Etherboot User Manual

Ken Yap

Markus Gutschke

This User Manual explains how to install, configure and compile the Etherboot package. The instructions here apply to version 5.2 of Etherboot.

1. About this User Manual

1.1. Obtaining the most recent version of this document

This document and related documents are also kept online at the Etherboot Home Page (http://www.etherboot.org/). This will in general have the latest source distributions and documentation.

1.2. Feedback

Comments on and corrections for this User Manual may be directed to the primary author (mailto:ken_yap AT users PERIOD sourceforge PERIOD net).

1.3. Copyrights and Trademarks

This manual may be reproduced in whole or in part, without fee, subject to the following restrictions:

- The copyright notice above and this permission notice must be preserved complete on all complete or partial copies.
- Any translation or derived work must be approved by the author in writing before distribution.
- If you distribute this work in part, instructions for obtaining the complete version of this manual must be included, and a means for obtaining a complete version provided.
- Small portions may be reproduced as illustrations for reviews or quotes in other works without this permission notice if proper citation is given. Exceptions to these rules may be granted for academic purposes: Write to the author and ask. These restrictions are here to protect us as authors, not to restrict you as learners and educators.

All trademarks mentioned in this document belong to their respective owners.

1.4. Acknowledgments and Thanks

Thanks to Markus Gutschke who wrote the first version of this document, and to all the people who have contributed information and corrections to this document.

For a list of people who have contributed substantially to the code, please see Section 10 section.

2. Introduction to Etherboot

2.1. What is Etherboot?

Etherboot is a software package for creating ROM images that can download code over an Ethernet network to be executed on a computer. Many network adapters have a socket where a ROM chip can be installed. Etherboot is code that can be put in such a ROM. Etherboot can also be booted from floppies (mainly for testing purposes but some people have been known to use this all the time) and hard disk, as a LILO/SYSLINUX compatible image, or from a hard disk partition, or via PXE.

Etherboot works on the x86, Itanium and Hammer architectures. It has not been ported to other platforms yet. Typically the computer is diskless and the code is Linux or FreeBSD, but these are not the only possibilities. The code uses the DHCP, tftp and NFS Internet Protocols, amongst others.

For a rather out of date but short talk/tutorial type introduction to what Etherboot does, see my SLUG talk (../diskless/t1.html).

2.2. License

Etherboot is Open Source under the GNU General Public License Version 2 (GPL2). The license conditions can be obtained from the file COPYING in the top directory of the distribution or viewed at www.gnu.org (http://www.gnu.org/). In particular note that if you are distributing binaries generated from Etherboot, such as ROMs or ROM images, you must provide or promise to provide the user with the source. If you have not made private changes to the code, you can do this by pointing the user to the Etherboot home page (http://www.etherboot.org/), noting the release version of course. If you have made private changes to the code, then you must distribute those changes too with binary distributions.

The following is not legal advice and you should seek your own legal advice but it is a reasonable interpretation of the GPL with respect to Etherboot and BIOS code. Installing Etherboot, either as an extension BIOS or combined in the BIOS chip is mere aggregation. The GPL does not extend to other works that a GPLed binary is aggregated with on a storage medium. The rationale is this: The BIOS does not need an Etherboot ROM to function and the Etherboot ROM can work equally well with another BIOS implementation. Therefore putting Etherboot either on a ROM chip or in the same chip as the BIOS does not cause the two to become a combined work. Under this interpretation, there is no fear that you have to GPL your BIOS if you ship Etherboot with your BIOS.

The GPL applies to the whole package but a few files may be used under other licenses for historical reasons. See Section 11 for details. Please support Open Source by joining the community and sharing. See the Etherboot home page (http://www.etherboot.org/) for some ways you can help Etherboot.

No Warranty or Support: Etherboot comes with NO warranties of any kind. It is hoped that it will be useful to you, but NO responsibility is accepted for any outcome of using it. Etherboot also comes with NO support, although you may get helpful advice from the mailing lists listed on the Etherboot home page.

2.3. What hardware is supported?

See Appendix A for a list of supported NICs. All Etherboot drivers are autoprobing, which means they attempt to detect the hardware addresses at which the card is installed. It's fairly easy to write a driver if you know C and are familiar with Ethernet hardware interfacing. Please read the developer manual (../devman/t1.html) if you wish to do so.

2.4. Getting help

Please join the Etherboot mailing lists (http://sourceforge.net/projects/etherboot). These are listed on the Etherboot home page. There is a users mailing list and a developers mailing list. The users list is for issues with building and running the software, while the developers list is for issues with features and coding.

Please post questions or bug reports to the Etherboot mailing lists, please do not mail me, because: 1. you get the benefit of a lot of experts seeing your question (no, I don't know everything, if only because there are many configurations I have never used); 2. a lot of people see the question and answer and this helps them too; 3. I have other demands on my time, like a job, and answering individual email is an unsustainable practice. You will probably not get any reply from me if you email me directly. I want to make the best use of my time and that is by making sure that as many people as possible see the questions and answers. Note that I will post my replies to the mailing lists so to see the answers you should subscribe, or be willing to check the archives later.

On the other hand, if you have a code or document contribution, then email to me is definitely appropriate. Please ask first before you send large files. Diffs are preferred to entire archives. If you are really keen to volunteer, and have the skills, ask to join the developer team.

Other lists you can join are the Linux Terminal Server Project mailing lists at LTSP (http://www.ltsp.org/). The LTSP list is focused more on the LTSP packages. However there is a fair amount of overlap between the lists and many key people are on all lists.

3. Unpacking, compiling and testing the package

3.1. A short cut to getting Etherboot images

Marty Connor has set up a web form (http://rom-o-matic.net/) for creating an Etherboot image on the fly and returning it as the output of the form. If all you want is an Etherboot image, this could save you having to build the distribution.

3.2. Unpacking the distribution

Unpack the distribution using gunzip and tar, using one of the following commands, where you replace x by the patchlevel number:

```
tar zxvf etherboot-5.2.x.tar.gz
tar jxvf etherboot-5.2.x.tar.bz2
gunzip < etherboot-5.2.x.tar.gz | tar xvf -
bunzip2 < etherboot-5.2.x.tar.bz2 | tar xvf -
```

If the documentation tarball was provided separately, then in addition do this:

```
cd etherboot-5.2.x
```

followed by one of the following:

```
tar zxvf ../etherboot-doc-5.2.x.tar.gz
tar jxvf ../etherboot-doc-5.2.x.tar.bz2
gunzip < ../etherboot-doc-5.2.x.tar.gz | tar xvf -
bunzip2 < ../etherboot-doc-5.2.x.tar.bz2 | tar xvf -
```

which will extract the documentation in a subdirectory of the Etherboot top directory.

3.3. Making an Etherboot image

To build an Etherboot image you need a recent release of gcc and the binutils tools. This package has been compiled with the tools from a SuSE 8.2 distribution but it should work with any recent Linux or FreeBSD distribution. gas 2.9.1 is too old to handle the 16-bit code in loader.S. Use gas 2.9.5 at least. Also the "gcc 2.96" used in RedHat 7.0 (and later versions maybe) generates faulty machine code compiling Etherboot. Use kgcc from those distributions instead.

Go to src/, edit the options in Config and make the image that you want. A full list of options is in Appendix B. We suggest you accept the default options for now if you are not sure what to select. Unlike 5.0, 5.2 does not make all the images by default, you have to explicitly state which ones you want.

You can test the image with a floppy before programming a ROM. On Linux just put a blank floppy in fd0 and say

```
make bin/card.zfd0
```

where *card* is the name of your network card and it will copy a bootable image onto the floppy. If you wish to do this by hand, perhaps because your floppy drive is elsewhere, just make bin/ *card*.zdsk and copy this binary to the floppy raw, i.e. starting at the boot block.

```
cat bin/3c509.zdsk > /dev/fd0
```

You can also use utilities such as rawrite to write the image onto the floppy.

Make sure the floppy has no bad blocks. It is best if it has been formatted just before use. You do not need to put any kind of filesystem on it. If you wish, you could substitute /dev/fd0 with the actual device suitable for the floppy size you are using, for example /dev/fd0H1440 for 1.44 MB floppies. This may be more reliable than using the autodetecting device /dev/fd0.

When you boot with this floppy it will load the Etherboot image from floppy and execute it. If you chose the correct image, it should be able to detect your card. To get the bootrom to acquire an IP address and load the intended code, you need to set up DHCP, tftp and NFS services, which we will discuss in the next section. We suggest you continue to use floppy booting until you have completed the setup of the server and are satisfied that diskless booting works.

In addition, you can generate images with the suffixes .zlilo, .zpxe, .com, and .zrom by saying:

```
 make bin/3c509.zlilo
```

and so forth.

The ones ending in .zlilo look sufficiently like Linux kernel images to be accepted by LILO, GRUB and SYSLINUX for installation. Unfortunately loadlin uses a slightly different method of booting for Linux kernels from LILO and SYSLINUX and will not work with these images.

The fact that .zlilo images look like a Linux kernel to LILO and SYSLINUX allows some interesting booting possibilities. For example, you could use LILO to select between DOS/Windows and Etherboot images from a disk that contains no Linux partitions, only FAT based partitions. This HOWTO (../lilo+etherboot/t1.html) shows you how this can be arranged.

The ones ending in .zpxe can be booted by a PXE booter. This is useful to chain to Etherboot from PXE. Here (http://www.geocrawler.com/lists/3/SourceForge/5299/125/6129709/) are some notes on how to combine PXE and Etherboot.

The ones ending in .com are DOS format executables, suitable for starting from DOS. It requires a real DOS environment, not a virtual DOS environment such as that provided by the DOS prompt window under Windows. Also it requires that there be no XMS drivers or other memory handlers loaded. It is not guaranteed to work if the environment is not clean, and sometimes not even if it is. The best chance of this format working is when DOS is booted with no device drivers whatsoever. If you can, use raw floppy or an intermediate bootloader for booting instead.

The ones ending in .zrom are images suitable for writing onto ROMs. If you are making a .zrom image, you must set the PCI vendor and device IDs correctly for PCI NICs. Look at the file NIC. Locate the line that has the correct PCI IDs for your NIC. This will give you the name of the ROM image you should use. The PCI IDs are usually displayed by the BIOS on booting up. They can also be read out from a running Linux system using the Linux PCI Utilities (http://atrey.karlin.mff.cuni.cz/~mj/pciutils.shtml). If you do not use the ROM with the correct IDs, the floppy version will work, but the ROM will not since the BIOS will check for a match.

There are also .fd0, .dsk, .lilo, .pxe and .rom counterparts to the .z* versions of the images. The difference is the .z* versions are compressed. Unless you doubt the (de)compression process, there is usually no reason to use the uncompressed versions..

4. Setting up a diskless boot

In this section I assume you want to boot a Linux kernel. Booting a FreeBSD kernel is documented elsewhere and does not require a generating a boot image. Booting a DOS kernel is similar, the main differences being in the way you set up the boot image.

4.1. Making a boot image

Etherboot expects to download a boot image in either ELF or tagged (../devman/t1.html) format containing the code to be executed. Briefly explained, a boot image has a wrapper around the pieces of code or data that need to be put in various places in the computer's memory. It contains a directory telling how large the pieces are and where they go in memory. It also says where to start execution.

A boot image is created using a utility program. The utility program (../mknbi.html) is specific to the kernel you want to load. The version for Linux is called mkelf-linux or mknbi-linux and that for DOS is mknbi-dos. These utilities are distributed separately and can be obtained from Etherboot web site (http://sourceforge.net/projects/etherboot/).

4.2. Compiling a custom kernel

The preferred method of running applications on the booted machine is to package an initial ramdisk (initrd) along with the kernel. This initrd can either provide all the files necessary for running the application (at a cost of using RAM to hold the files) or mount NFS filesystems to obtain other files. Therefore you will need to compile ramdisk options in your kernel. For an example of how an initrd is created and packaged, see the LTSP project sources.

After you have compiled the kernel, make the boot image, like this:

```
mkelf-linux --output=/tftpdir/xterm.nb zImage initrd.gz
```

This puts the image in where the tftp daemon expects to find it, in this example /tftpdir. Make sure it is world-readable because typically the tftp daemon runs as an unprivileged user. It is recommended that you set a path explicitly for tftpd instead of relying on any defaults. For example, for inetd, the entry in /etc/inetd.conf looks like this (xinetd uses a different syntax, check the documentation):

```
tftp dgram udp wait root /usr/sbin/tcpd in.tftpd /tftpdir
```

4.3. Setting up a DHCP daemon

You need to set up a DHCP server to hand out an IP address and other configuration information to the client.

The main requirement of the DHCP server is that it needs to send out suitable configuration information. Prior to version 5.0.7, Etherboot accepted any DHCP offer (but see REQUIRE_VCI_ETHERBOOT below for an exception). Since version 5.0.7 Etherboot will not accept any DHCP offer where the server IP is empty (all zero) or the filename is empty. These offers are useless to Etherboot anyway so ignoring these offers will give it a better chance of picking the right DHCP server.

If you already have a DHCP server on your network for providing Windows clients with IP addresses, chances are that it is not a suitable DHCP server because it's only tailored to the single purpose of handing out client addresses. Suitable DHCP servers include the ISC DHCPD, available for most Unix/Linux systems. You can run such a DHCP server in parallel with the Windows one, provided you do not attempt to give Windows clients leases, in which case there would be a clash. You can exclude Windows clients in two ways. One is to register the only the MAC addresses of the diskless clients in

/etc/dhcpd.conf, and to make sure that the server is declared "not authoritative". The second is to look for the Vendor Class Identifier of "Etherboot-5.x" in the DHCP discover packet.

If you already have a DHCP server on your network that does provide the server IP and the filename, then you have to either use that DHCP server by editing its configuration file. This may require the cooperation of the system admin if you are not the admin. If you are not able to configure the DHCP server, then proceed to the section on REQUIRE_VCI_ETHERBOOT.

The minimum information you need to put in /etc/dhcpd.conf is:

1. The domain name of the machine.

2. The Ethernet (MAC) address of the network card, which you generally obtain from a sticker on the card, a configuration program for the card, or in the last resort, from watching the output of Etherboot or from the packets sent from the card when trying to boot, using the debug option of DHCPD.

3. The name of the boot image file, relative to the tftpdir directory.

4. The IP address you intend to give it, or the dynamic range it is to come from.

5. The TFTP server defaults to the DHCP server if not specified with next-server.

Here is a sample DHCP configuration for ISC dhcpd:

```
option domain-name "ken.net.au";
option domain-name-servers 192.168.0.1;
option broadcast-address 192.168.0.255;
use-host-decl-names on;
subnet 192.168.0.0 netmask 255.255.255.0 {
 host xterm {
  hardware ethernet 08:00:2B:B7:F3:80;
  fixed-address xterm.ken.net.au;
  filename "/tftpdir/xterm.nb";
 }
}
```

The declaration "use-host-decl-names on" tells dhcpd to include the name xterm in the reply so that it can be used as part of pathname to mount by NFS, etc.

If your tftp server is not the same as the DHCP server, use the next-server declaration to specify a tftp server.

The 2.4.4 and above kernels can do a separate DHCP request to obtain an address. Kernels in the 2.2 series use BOOTP. However user space DHCP configuration is now preferred.

More information about DHCP can be found at the DHCP FAQ (http://www.dhcp-handbook.com/dhcp_faq.html).

If you are on a local network that is not directly connected to the Internet, you can use the "private" IP addresses 192.168.x.y (or in the other ranges mentioned in RFC1918 (http://www.ietf.org/rfc/rfc1918.txt)). Otherwise please ask either your network administrator or your Internet service provider for your own IP address(es).

You could use a bootp daemon instead of a DHCP daemon. However DHCP is preferred as DHCP provides more information to control the boot process. Also there is a bug in some old versions of bootpd where unnecessarily large reply packets are sent, causing packet fragmentation at the UDP level, and dropping of the packets by Etherboot. DHCPD can emulate bootpd if that's what you really want.

4.3.1. REQUIRE_VCI_ETHERBOOT

You may need to Etherboot in an environment where you have to install your own DHCP server, and it may interfere with the existing DHCP server. Etherboot has a feature designed to overcome this problem. It consists of two parts:

The first part is Etherboot contains code that when it sends out a DHCPDISCOVER request, a tag containing a Vendor Class Identifier of "Etherboot-x.y" is sent out, where x.y is the version number, currently 5.2. The 5 and the 2 are printable digits, not binary values, i.e. byte values 53 and 50 respectively. This allows the server to identify Etherboot clients and ignore all others.

The second part is Etherboot can be conditionally compiled to require a Vendor Encapsulated Option containing a VCI of "Etherboot", otherwise it will ignore the DHCPOFFER. The option is not compiled in by default because it would cause Etherboot to not boot in plain setups. The server we want to respond will include this tag in DHCPOFFERs and while other servers (e.g. Windows servers) may respond, they will be ignored.

Vendor Encapsulated Option is supported in ISC DHCPD 2 or 3. (It's not documented in DHCPD 2, but it works.) Other DHCP servers may support VEO. (It's a RFC2132 option.) In ISC DHCPD 3, follow the documentation for declaring a VEO scope. In ISC DHCPD 2 the magic line required is:

```
option vendor-encapsulated-options 3c:09:45:74:68:65:72:62:6f:6f:74:ff;
```

Put this in the appropriate scope. Translation: Vendor Encapsulation Options (Option 43) encloses: Vendor Class Identifier (Option 60 = 0x3c), length 9, value "Etherboot"; End of Options (Option 255 = 0xff).

If you have a DHCP server already running for the subnet, you probably should specify that your additional ISC DHCPD server is not authoritative for the the subnet, or it will veto (NAK) existing client IP address allocations, upsetting the status quo. See the ISC DHCPD options documentation.

Here is a practical document (http://www.geocrawler.com/archives/3/5299/2001/7/100/6129709/) describing how it is done.

4.4. Setting up a tftp daemon

Now set up a tftp daemon. This means installing the tftp package and making sure that the tftp service is active in /etc/inetd.conf. If you want to be very careful you may wish to use the secure (-s) option of tftpd, this chroots to the specified directory, but then your pathnames in /etc/dhcpd.conf must be relative to the new root directory.

If you are booting many clients you should be aware of the limitations of running tftpd from inetd. Typically inetd has limits on how often a daemon can be spawned, to detect runaway daemons. If many clients request the tftp service within a short period, inetd may shutdown that service. If you have a setup where there are many clients, it may be better to use an improved replacement for inetd, such as xinetd.

Another possibility is to install a better tftp daemon like atftp at the Etherboot web site. This one can be run as a standalone daemon, avoiding *inetd limitations and it multithreads internally.

5. Testing the network booting

Now when you start up Etherboot, it should obtain an IP address and print out what it received. If you do not get this to work, turn on debugging in DHCPD and see if any query was received. You may also wish to use the tcpdump or ethereal utilities to watch the network for DHCP packets (port bootps). If not, check your network hardware (cables, etc). If a query was received, check if DHCPD was able to give an answer. If not, then the Ethernet address was not found in `/etc/dhcpd.conf`. If a reply was sent, then only faulty hardware or a bug in Etherboot would prevent it being received by Etherboot.

Assuming an IP address was received, the next thing Etherboot tries to do is load a file using tftp. Check your system logs to see if a tftp daemon was started up and a file requested. Generally if you run tftpd under tcpwrapper security, a log entry will be generated. If not, it could be a path problem or file permission problem (the file needs to be readable by tftpd). Another problem could be that tftpd needs to reverse map the IP address to a name for security checking, and you don't have the client's details in `/etc/hosts` or in DNS, or your tcpwrapper config files (`/etc/hosts.deny`, `/etc/hosts/allow`) do not allow the access. Fix the problem.

After the boot image is loaded, Etherboot will jump to it. If it crashes here, check that the image is a boot image. If it executes and stops at the point where it's trying to mount NFS filesystems, then you need to check your NFS mounts. Another common problem is the shared libraries on the NFS partition are not suitable for your CPU, e.g. i586 libraries but a 486 diskless client.

5.1. Setting up an initrd filesystem

Recent advances in the Linux kernel (2.4 and above) have made the use of an initrd that does user space autoconfiguration and mounting of a NFS root filesystem, followed by a pivot_root, a more flexible alternative to kernel autoconfiguration and mounting of a NFS root filesystem. Postings on the kernel mailing lists indicate that at some point in the future, kernel level autoconfiguration (BOOTP/DHCP from the kernel) may be removed from the Linux kernel and initrds will be the recommended way to start up a diskless system.

Until I have time to write detailed instructions on how to construct the initrd, I refer you to the LTSP distribution (http://www.ltsp.org/) which uses this technique. Mknbi supports initrds, see the ramdisk argument of mkelf-linux (../mknbi.html). The Linux kernel documentation describes the extra arguments should be passed to the kernel to make it use an initrd, and how to arrange the initrd so that the startup script within it is called when it's mounted. If the initrd mounts a NFS root filesystem then it should still have all the needed structure as explained in the next section.

Initrds can also be used for mounting other network filesystems instead of NFS root. Some applications could even run totally out of initrd, e.g. Floppy Firewall (http://www.zelow.no/floppyfw/), provided you have the memory, of course.

5.2. Other filesystem setups

This tutorial does not cover all possible ways of setting up a diskless client's initial filesystem. You could even mount a conventional hard disk. Why would you want to boot "diskless" if you have a hard disk? Reasons might be: you do not wish to administer the local disk; you want the assurance that a system is running a kernel from a central server; or you like the speed of network booting. Network booting is one technique in a toolbox. Techniques can be combined to do what you want. If you are interested in running the diskless client as an X-terminal, a very common use, you may wish to investigate the Linux Terminal Server Project (http://www.ltsp.org/).

5.3. Swap over NFS

Swap over NFS can be arranged but you have to patch the kernel source. See here (http://nfs-swap.dot-heine.de).

Be aware that opinions are divided on NFS swap. Some people think it's a bad thing because it just kills the network if you have lots of diskless computers and that you shouldn't be running into a swap regime on a diskless computer anyway. Some other people like having a bit of insurance.

Also have a look at the NBD (http://atrey.karlin.mff.cuni.cz/~pavel/nbd/nbd.html) Network Block Device for swapping over that.

There is also the follow-on project ENBD (http://www.it.uc3m.es/~ptb/nbd/), the Enhanced Network Block Driver. I have no experience with this for swapping. Comments welcome.

6. Booting DOS

What about DOS? The deal with DOS is that one is loading a virtual floppy called A: into extended memory and then booting from this floppy. So you have to capture an image of a bootable DOS floppy first. Some more details can be found in the mknbi-dos (../mknbi.html) utility.

I have booted DOS (both M$ versions up to 5.0 and DR versions up to 7.03) diskless this way. A mknbi-fdos (../mknbi.html) is available for building boot images for booting FreeDOS, the procedure differs slightly from booting M$ or DR DOS.

If you were thinking of booting a Windows machine via the network, it seems (I'm not masochistic enough to do this) the problem is not the network booting but the mounting of a file system over NetBIOS (Windows does not do remote mounts of root filesystems over NetBIOS on TCP). So that rules out a Samba server. It appears to be possible over a Netware server, for which Linux or FreeBSD has workalikes. Also it is said that only certain versions of Windows will allow diskless booting. You will also have problems with pathnames and the usual Windows hassles. Do you really want to do this? You do know that you can run lots of desktop applications like Netscape, StarOffice, etc. on Linux, FreeBSD, etc. now? Why pay good money when you can use equally good free replacements? Anyway if you are still determined, in the Etherboot home page (http://www.etherboot.org/), there are links to external Web pages, one explaining how this was done with a commercial TCP/IP boot ROM, another explaining how to do it using Etherboot and Netbios over IPX. A recent user experience is here (http://www.geocrawler.com/lists/3/SourceForge/5299/0/8052616/). Good luck and send us your experiences or better still a URL to a page explaining how you did it.

7. Making an Etherboot EPROM or EEPROM

Assuming you have satisfactorily set up your server environment, you may now wish to put the Etherboot onto an EPROM or EEPROM. Naturally this assumes access to hardware to program (and possibly erase) EPROMs. Access to a friendly electronics engineer and/or lab is one way to program and erase EPROMs. Otherwise you can look at the commercial links page (http://www.etherboot.org/clinks.html) for places you can buy programmed EPROMs from.

Some high-end network cards, for example the 3Com 905B, have a socket for an EEPROM which can be programmed in situ with the right utilities. See any release notes accompanying Etherboot for more information.

7.1. Choosing the EPROM

Most network cards come with a blank (E)EPROM socket even though it is seldom used. When it is used, it is typically filled with a proprietary EPROM from the network card manufacturer. You can put an Etherboot EPROM there instead.

7.2. Enabling the EPROM

First you must discover how to enable the EPROM socket on your card. Typically the EPROM is not enabled from the factory and a jumper or a software configuration program is used to enable it.

7.3. Size and speed of the EPROM

Secondly, you must discover what size and speed of EPROM is needed. This can be difficult as network card manufacturers often neglect to provide this information.

The smallest EPROM that is accepted by network cards is an 8k EPROM (2764). 16kB (27128), 32kB (27256), 64kB (27512) or even 128kB (27010) EPROMs are possible. (You will often see a C after the 27, e.g. 27C256. This indicates a CMOS EPROM, which is equivalent to the non-C version and is a good thing because of lower power consumption.) You want to use the smallest EPROM you can so that you don't take up more of the upper memory area than needed as other extensions BIOSes may need the space. However you also want to get a good price for the EPROM. Currently the 32kB and 64kB EPROMs (27256 and 27512) seem to be the cheapest per unit. Smaller EPROMs appear to be more expensive because they are out of mainstream production.

If you cannot find out from the documentation what capacity of EPROM your card takes, for ISA NICs only, you could do it by trial and error. (PCI NICs do not enable the EPROM until the BIOS tells the NIC to.) Take a ROM with some data on it (say a character generator ROM) and plug it into the socket. Be careful not to use an extension BIOS for this test because it may be detected and activated and prevent you from booting your computer. Using the debug program under DOS, dump various regions of the memory space. Say you discover that you can see the data in a memory window from CC00:0 to CC00:3FFF (= 4000 hex = 16384 decimal locations). This indicates that a 16kB EPROM is needed. However if you see an alias in parts of the memory space, say the region from CC00:0 to CC00:1FFF is duplicated in CC00:2000 to CC00:3FFF, then you have put an 8kB EPROM into a 16kB slot and you need to try a larger EPROM.

Note that because pinouts for 28 pin EPROMs are upward compatible after a fashion, you can probably use a larger capacity EPROM in a slot intended for a smaller one. The higher address lines will probably be held high so you will need to program the image in the upper half or upper quarter of the larger EPROM, as the case may be. However you should double check the voltages on the pins armed with data sheet and a meter because CMOS EPROMs don't like floating pins.

If the ROM is larger than the size of the image, for example, a 32 kB ROM containing a 16 kB image, then you can put the image in either half of the ROM. You will sometimes see advice to put two copies of the image in the ROM. This will work but is not recommended because the ROM will be activated twice if it's a legacy ROM and may not work at all if it's a PCI/PnP ROM. It is tolerated by Etherboot because the code checks to see if it's been activated already and the second activation will do nothing. The recommended method is to fill the unused half with blank data. All ones data is recommended because it is the natural state of the EPROM and involves less work for the PROM programmer. Here is a Unix command line that will generate 16384 bytes of 0xFF and combine it with a 16 kB ROM into a 32 kB image for your PROM programmer.

```
 (perl -e 'print "\xFF" x 16384'; cat bin/3c509.zrom) > 32kbimage
```

The speed of the EPROM needed depends on how it is connected to the computer bus. If the EPROM is directly connected to the computer bus, as in the case of many cheap NE2000 clones, then you will probably have to get an EPROM that is at least as fast as the ROMs used for the main BIOS. This is typically 120-150 ns. Some network cards mediate access to the EPROM via circuitry and this may insert wait states so that slower EPROMs can be used. Incidentally the slowness of the EPROM doesn't affect Etherboot execution speed much because Etherboot copies itself to RAM before executing.

If you have your own EPROM programming hardware, there is a nice collection of EPROM file format conversion utilities here (http://srecord.sourceforge.net/index.html). The files produced by the Etherboot build process are plain binary. A simple binary to Intel hex format converter can be found at the Etherboot web site here (http://www.etherboot.org/bin2intelhex.c).

Etherboot is believed to make PnP compliant ROMs for PCI NICs. A long-standing bug in the headers has been tracked down. However some faulty old BIOSes are out there so I have written a Perl script swapdevids.pl to switch the header around if necessary. You'll have to experiment with it both ways to find out which works. Or you could dump a ROM image that works (e.g. RPL, PXE ROM) using the Perl script disrom.pl. The fields to look at are Device (base, sub, interface) Type. It should be 02 00 00, but some BIOSes want 00 00 02 due to ambiguity in the original specification.

8. Troubleshooting tips

- Floppy boot doesn't work. Is the floppy error-free? Have you copied the ROM image (with the disk loader prepended) to the floppy raw? Is that size of floppy bootable by your computer? Are you trying to run Etherboot on a 16 bit machine (286, 086/088)? Have you selected too many compile time options? The real limit on Etherboot is not the size of the EPROM but the fact that it executes in the 48kB region between 0x94000 and 0xA0000. If the sum of code, stack and bss is greater than 48kB, then Etherboot might crash at unexpected places.

- Floppy version works but EPROM version doesn't work. There is a program called rom-scan (Linux, FreeBSD and DOS versions) in the directory contrib/rom-scan which will help detect problems. Rom-scan will only work on ISA ROMs though.

  - If the EPROM is not detected at all then the contents of the EPROM are not visible to the BIOS. Check that you have enabled the EPROM with any jumpers or soft configuration settings. Check that you do not have any conflicts in the memory address of the EPROM and any other hardware. Perhaps you have to prevent it from being mapped out by your BIOS settings. Or perhaps you have to shadow it with RAM. Maybe you put the code in the wrong half or wrong quarter of the EPROM. Maybe the access time of the EPROM is not low enough. You can also use the debug program under BIOS to examine the memory area in question.

  - If rom-scan says the EPROM is present but not active, then something prevented the BIOS from seeing it as a valid extension BIOS. This could be truncation of the EPROM window, maybe you have a larger EPROM in a slot meant for a smaller one. Maybe there is a checksum error in the EPROM due to some bits not properly programmed or the EPROM not being fast enough. In one case that we know of, the 3c503 network card, the ASIC actually returns 2 bytes of 80 80 in the end locations of the EPROM space. This apparently is a kind of signature. The makerom program in Etherboot compensates for this, but if the pattern is not 80 80, then the code needs to be modified.

  - If rom-scan says the EPROM is present and active, but BIOS does not see it, then perhaps the EPROM is located in an area that the BIOS does not scan. The range scanned is supposed to be 0xC0000 to 0xEF800 in increments of 2kB but I have seen some BIOSes that continue the scan into the 0xF0000 page.

  Note that rom-scan will also detect other extension BIOSes mounted on your computer, for example VGA BIOSes and SCSI adapter BIOSes. This is normal.

  For PCI NICs there may be a different reason why the ROM does not work. The PCI IDs of the ROM must match those of the NIC controller chip or the BIOS will ignore the ROM. The floppy version does not undergo this check since it isn't directly called from the BIOS. You must use the ROM image with the correct PCI IDs for your NIC.

- Etherboot does not detect card. Are you using the right ROM image? Is the card properly seated in the computer? Can you see the card with other software? Are there any address conflicts with other hardware? Is the PCI id of the card one that is not known to Etherboot yet? In this case and where you think there is a bug in Etherboot, please submit a report to the Etherboot-users mailing list.

- Etherboot detects card but hangs computer after detection. Some cards are booby traps while they are enabled. The typical offenders are NE2000s which will hang the bus if any access is made to the reset addresses while interrupts are active. You may need to do a hard reset of the computer, i.e. power down and up again before running Etherboot.

- Etherboot detects card but does nothing after saying Searching for server. Check your network hardware. Did you select the right hardware interface (AUI, BNC, RJ45)? Is the cabling ok? If you have a Unix computer on the network and have root privileges, you could run tcpdump or ethereal looking for broadcast packets on the bootps port. If the requests are getting sent out but no replies are getting back, check your DHCPD setup. Also check if the server has a route to the client.

- Etherboot obtains IP address but fails to load file. Check the tftp server. Is the boot image file installed? Is the file world readable? Is the path to the file allowed by the configuration of tftpd? Is the client denied by tcpwrapper rules? Did you put the right home directory and filename in

`/etc/dhcpd.conf`? If you are booting lots of clients, is inetd shutting down tftpd for being spawned too often? If so, you need to get a better inetd or a a dedicated tftpd that runs as an independent daemon.

- Etherboot loads file but hangs halfway through the transfer. We have one report that this happens if the Fast A20 Gate option is selected in the BIOS setup. The A20 issue is explained here (http://www.win.tue.nl/~aeb/linux/kbd/A20.html). In effect this causes the loaded kernel to overwrite Etherboot and kill it. Try deselecting the option. You don't need it for Linux anyway.

- Etherboot loads file via tftp but Linux fails to mount the root filesystem, either on an initrd or on NFSroot. This is a large category. Here are some suggestions:

  - You do not have a private copy of the /, /etc, /var, ... directories

  - Your /dev directory is missing entries for /dev/zero and/or /dev/null or is sharing device entries from a server that uses different major and minor numbers (i.e. a server that is not running Linux).

  - Your /lib directory is missing libraries (most notably libc* and/or libm*) or does not have the loader files ld*.so*

  - You neglected to run ldconfig to update /etc/ldconfig.cache or you do not have a configuration file for ldconfig.

  - Your /etc/inittab and/or /etc/rc.d/* files have not been customised for the clients. For example if you set the wrong IP address in your client's init files you could interfere with the server.

  - Your kernel is missing some crucial compile-time feature (such as NFS filesystem support, booting from the net, transname (optional), ELF file support, networking support_.

  - Missing init executable (in one of the directories known by the kernel: /etc, /sbin, ?). Remember /sbin/init must be a real file, not a symlink.

  - Missing /etc/inittab

  - Missing /dev/tty?

  - Missing /bin/sh

  - System programs that insist on creating/writing to files outside of /var (mount and /etc/mtab* is the canonical example)

The essence is that you must provide whatever is needed in the root filesystem that your kernel needs to boot.

9. Frequently Answered Questions

Please check this section if you have a problem before asking on the mailing lists.

9.1. Building Etherboot

1. What do I need to build Etherboot?

You need gcc, gas and binutils, as well as any accompanying libraries and include files. Generally speaking on a package based system using RPM or DEB, you will need the C compiler package, the include file package, the C library package, the assembler package and the binutils package (this may include the assembler). You will also need perl if you modify the file NIC or use a different RELOCADDR value in the Makefile than the default, and m4 if you modify this file userman.xsgml.

2. I get an error from as saying data32 is an unknown directive or it has errors with assembler files.

    Your gas is too old, upgrade to 2.9.5.

3. I get warnings about ljmp *.

    They are harmless, you can ignore these. They are due to changes in the assembler syntax between gas versions. We could get rid of the warnings if we could easily detect which patches are installed in the version of gas you are using (it's not just a matter of detecting the gas version) but we'd rather just wait for the old gas versions to disappear since they are just warnings.

4. The documentation talks about mkelf-linux and mknbi-dos. Where are they?

    These are distributed from the Etherboot web site (http://sourceforge.net/projects/etherboot/).

5. Why don't you provide prebuilt ROM images?

    rom-o-matic.net (http://rom-o-matic.net) is the answer. This is a site that makes ROM images for you on demand from specifications given to a web form and returns the image as the result of the form.

9.2. Testing Etherboot

1. I put the ROM image on floppy like you wrote (cat bin/boot1a.bin bin/foo.rom > /dev/fd0, or make bin/foo.fd0) but the loader prints out an error.

    The floppy you use should be an error-free, preferably a recently formatted floppy. Do not trust new floppies; they have been known to lose their manufacturer formatting in storage. You don't need to put a filesystem of any sort on it, FAT or ext2 or otherwise. Another possible cause is that there are alignment differences between the drive used to write the floppy and that used on the target machine.

2. My network adapter is detected but I get no reply to the BOOTP/DHCP request.

    Do you have a BOOTP or DHCP server running on the same Ethernet segment? On many operating systems the server is not enabled by default. Review the instructions in the Troubleshooting section. Another thing to note is that the BOOTP (or DHCP in fixed address mode) server will not reply if it does not know the network adapter's Ethernet address. Since the address may be hard to determine if it is not printed on the card or you do not have the adapter's setup program, you can copy it from Etherboot's startup message. Did you also provide a filename with the DHCP offer? Offers with no

filenames are ignored by Etherboot. Remember to restart the server if you have edited the config file and the server does not automatically reread when it discovers an updated config file.

Another thing to check is that the BOOTP or DHCP server is allowed to receive the query. You may have some protection mechanism such as tcpwrappers or a firewall in front of it. As the booting computer does not have an IP address, the request will come from 0.0.0.0 so your rules must allow through packets from this address.

If the BOOTP or DHCP server is on another Ethernet segment, things get more complicated. You need to run a BOOTP or DHCP relay. You will probably also need to set the gateway field in the reply so that TFTP will work across the gateway. You should read a good explanation of how these work, in say, W. Richard Steven's book TCP/IP Illustrated.

3. Etherboot gets the BOOTP/DHCP parameters but cannot find a TFTP server.

Do you have a TFTP server installed and running and is it allowed to serve the client in question? For example the tcpwrapper rules may not allow TFTPD to respond to the IP address the booting computer is at. You should look at the log files on the server for any clues.

4. The TFTP server is found but it replies Access violation.

Access violation is a blanket reply for many different problems but essentially the TFTP server cannot give Etherboot the file requested. Did you put the file where TFTPD expects to find it, e.g. on a directory that is on its path? Did you make the file world readable? Case of the filename is important too. Check the log files on the TFTP server to see what the actual filename it tried to open was, sometimes directory prefixes are prepended to the name due to the program options specified.

5. I made this kernel and put it in /tftpdir like you wrote but Etherboot says Unable to load file.

Is the file a boot image? You cannot use a ordinary kernel image, you must process it with mkelf-linux (../mknbi.html) first.

6. I have this proprietary boot ROM (e.g. LanWorks, PXE, etc) and I used mkelf-linux or mknbi-linux to make a boot image, (or I got this boot image from the LTSP project), but the boot ROM doesn't load it, or it fails to run.

The boot image format is specific to Etherboot. It will not work with proprietary boot ROMs. You have to find out from the supplier what boot procedures you should use. For example, if you are using a LanWorks boot ROM, the information you need is here (http://www.3com.com/managedpc). For PXE the utility you need is PXELINUX.

9.3. Hardware capabilities

1. What network cards are supported?

See the list of supported NICs in the Appendix.

2. I have a machine with the X processor and Y megabytes of memory. What can I expect to run on it?

    Please note that these estimates are approximate:

    - On a 386 with at least 4MB of memory you can boot Linux. With 4MB perhaps only a few telnet sessions are possible. With 8MB you might be able to run a text based web browser like Lynx or W3M. You can also run firewalls such as floppyfw (http://www.zelow.no/floppyfw/).

    - On a 486 with 16MB of memory you can run X to make an X-terminal.

    - On a Pentium with 32MB of memory you can run an X-terminal and some applications locally, say perhaps printing, daemons to control devices, etc.

    - On anything faster and with more memory you could perhaps do distributed computation, e.g. a cluster.

9.4. Booting Linux

1. The kernel loads but it cannot find a NFS server for the filesystem.

    Do you have a NFS server running and is it allowed to serve this client? NFS is actually several services. On Linux at least you need: nfsd (either kernel or userland version), rpc.mountd and portmapper. Check if the tcpwrappers config file is allowing portmapper to receive the request. Look at the log files for clues. Did we already mention that log files are your friends?

2. The filesystem mounts but it says something about not being able to open an initial console. Or alternatively, various services complain about not being able to write to the filesystem.

    A common mistake in Linux NFS servers is to put extra spaces in /etc/exports. Please see the NFS FAQ (http://nfs.sourceforge.net/) for frequently answered questions about Linux NFS services.

    Please review the Troubleshooting section for what is required on this root filesystem. The situation is complicated by the fact that there are many possible ways of setting this up, including using a root filesystem that is on ramdisk. If you wish to avoid many of the troubles, try using a packaged solution such as LTSP (http://www.ltsp.org/).

9.5. Running X

1. I tried to run X on the client but it aborted.

Remember that the config files used by the X server should pertain to the client's video adapter and display hardware. If you used a LTSP (http://www.ltsp.org/) package, please review the configuration directions. If you used the copy files from server solution, then you need to customise the X server configuration. Another thing that may cause the server to abort is lack of a mouse device.

2. X -query server runs but all I get is a gray stippled screen.

Either you don't have an XDM server running on the server machine or it is not allowed to serve this client. In the latter case check XDM's Xaccess file, because for security reasons, the ability for clients to connect is usually disabled.

3. When I am logged in using an X-terminal, I find that the floppy drive, sound card and name of the computer are those of the server!?

This is to be expected! This is exactly how an X-terminal works. You are indeed logged onto the server and the client just provides display (screen) and input device (keyboard and mouse) services to the application. This is one of the beauties of the X Windowing system model, it's *network transparent*.

4. So how do I run applications on the client? I have this (smartcard reader, printer, sound card, etc) program that must execute locally.

The client can be configured to allow you to run programs locally. What you have to do is either start the application from the init scripts run during bootup, as would be the case for the printer and sound daemons mentioned later; or to start a shell interpreter on the client, either through an interactive login to the client or a remote execution of a command from the server to the client. Common methods are rsh (insecure) and ssh (better).

5. X applications cannot find (some) fonts.

Do you have an X font server (XFS) running on the server machine, is it allowed to serve this client, and has the client been told to use the font server? The last point is usually configured in the XF86Config file, or by a xset command to modify the font path after logging in.

Also note that RedHat (and possibly other distributions) has made XFS by default serve only the local machine using a Unix socket. You need to modify the startup script to tell XFS to use a TCP/IP socket.

6. How much CPU power and memory do I need on the client? On the server?

It depends on the configuration. There are two major cases: where the client is an X-terminal, and not much more; and where the client is configured to run applications locally.

An X server will fit in 16MB of memory, and 32MB is quite adequate. Performance depends on the CPU, video card and your expectations. An old Pentium 200 with a PCI video card does very well, but if you are not fussy, a high-end 486 with a VLB video card can be satisfying too.

If you want to run apps locally, well how long is a piece of string? Netscape will need say another 16MB. It all depends. Whatever you do, it's worth trimming down on the services you run on the client. Don't run more virtual consoles than you need and don't run unneeded daemons.

As for the server, in the X-terminal case this has all the applications running on it, so it should be adequate for the multiuser aspect. A high-end Pentium, with 64 MB of memory to start with, and between 8 and 16MB for each extra client is a good starting point. It will also depend on your mix of client access, statistically perhaps not everybody will be running at the same time. Remember that you don't have to have one big server for all your clients, you can and you should distribute the load across servers.

9.6. Other client applications

1. How can I print to a printer attached to a diskless client?

   There is a server program called p910nd (http://www.etherboot.org/p910nd/) at the Etherboot web site that funnels data from a TCP/IP connection to the printer port. You can instruct lpd or CUPS (http://www.cups.org/) on the server to send jobs across the network to p910nd.

2. How can I output sound on the client?

   There is a package called virtualfs (http://www.solucorp.qc.ca) that proxies the sound devices across the network. It can also proxy the floppy drive.

   Another solution is EsounD (http://www.tux.org/~ricdude/EsounD.html)

   You may wish to check the LTSP (http://www.ltsp.org/) site and mailing lists for other proposed solutions.

3. How can I access the floppy on the client?

   Besides virtualfs mentioned above, recent distributions of mtools (http://wauug.erols.com/pub/knaff/mtools/) have a floppyd. This only works with the mtools utilities though.

9.7. Booting FreeBSD

1. Where are the instructions for booting FreeBSD?

   For now, there is just a short document in the doc directory. Better versions of this document depend on contributions from the FreeBSD community, I am unable to test FreeBSD because I don't run it.

9.8. Booting other operating systems (DOS, Windows)

1. I want to boot FreeDOS.

   The new mknbi utility supports creating boot images from FreeDOS kernels now. See the man page (../mknbi.html) for details. FreeDOS is under development and if the layout of the kernel image changes, please send any corrections to me.

2. I cannot boot DR-DOS 7.03.

   There is some difference between the DR-DOS 7.03 and 7.02 bootblock that causes it not to boot. But a 7.02 bootblock works just as well with the DR-DOS 7.03 kernel, so you can substitute that.

3. DOS dies when I load HIMEM.SYS.

   Use the /testmem:off option to prevent HIMEM from scribbling over the ramdisk which is the floppy A:.

4. How do I make A: my real floppy again after booting is complete?

   Use the rmrd.com program supplied with mknbi (../mknbi.html).

5. I want to use the real floppy at the same time I am using the ramdisk image of the boot floppy.

   The --harddisk option of mknbi-dos is intended for this. It causes your boot drive to be C:, so you can use A: for the real floppy. See the man page (../mknbi.html) for more details

6. I want to boot Windows.

   I pass on this one, as I do not have (by choice) any Windows systems running on my computers. Perhaps others can contribute to this section. However I gather that it is only possible on Windows95A, as other versions don't have the necessary support for diskless booting.

9.9. Hardware issues

1. Where can I get an EPROM made?

   Depending on where you live, you might find a supplier listed on the Commercial Links page. Another possibility is to seek the help of someone working in a university or industrial lab who has an EPROM programmer. If you are handy with hardware, you could buy a kit or build your own. There are links to kit suppliers in the Commercial Links part of the home page.

   Some high end adapters, for example the 3Com and Intel ones, accept an EEPROM in the socket. This can be programmed in-situ using utility programs, some of which or information about are under the contrib directory in the Etherboot distribution.

Finally some recent motherboard have flash BIOSes which contain space where an extension BIOS such as Etherboot can be inserted. The Phoenix Award BIOSes can be modified using a program called cbrom.exe possibly here (http://www.ping.be/bios). Or do a Web search for it. No success has been reported for AMI BIOSes. Dirk von Suchodoletz maintains a list of successes and failures here (http://goe.net/anleitungen/award_board.html).

Here is some text contributed by Dirk von Suchodoletz. He hopes to put it on a web site someday:

```
6.4 Using your mainboard's BIOS to integrate etherboot-code

Newer mainboards that have an AWARD-BIOS can use etherboot without
separate EPROMS and therefore without the necessity of having a
EPROM-programmer[?]. (Heinrich Rebehn wishes to add: Flashing the BIOS
is always a (small) risk. Flashing with unsupported, hacked BIOS image
is *dangerous* and may render your PC unbootable. If you don't
have access to a PROM burner you should stay away from experimenting.) In
order to do this, you need 2 software tools: awdflash.exe, which should be
included in your mainboard package, and cbrom.exe, which is an OEM-tool
that allows modifications of the BIOS. awdflash.exe reads and writes
the flashrom content, whereas cbrom.exe is used to analyse the content
of the AWARD BIOS image. cbrom.exe can also add code to the BIOS image
or remove components. This way you can easily integrate etherboot into
your mainboards without even opening the PC's case.

After the BIOS image has been saved (e.g. as bios.bin), or in case the
current version of the BIOS has been copied from the board manufacturer's
website, 'cbrom bios.bin /d' shows how much space is left on the image
for your code.

As the flashrom holds the compressed BIOS, cbrom will also compress
the code when adding it to the BIOS. Therefore, 8 to 20 kbyte of free
memory is needed, depending on the network adapter's driver. In case not
enough memory is left, unneeded BIOS components can be removed from the
BIOS image to regain space: the manufacturer's logo or the Symbios/NCR
SCSI-code are note needed for diskless systems. 'cbrom bios.bin
/[pci|ncr|logo|isa] release' will remove those unnecessary components.

The command line "cbrom bios.bin /[pci|isa] bootimg.rom [D000:0]" adds
the compiled etherboot code to the bios. bootimg.rom is the code we
would use to burn onto EEPROMs in other cases. Depending on your network
card, either the pci or isa options have to be used. With isa cards
you have to tell cbrom to which RAM location the code will be extracted
at boot time. Attention: Compile the etherboot with the -DASK_BOOT or
-DEMERGENCYDISKBOOT option to be able to access a disk. The code added
by cbrom will be executed before the computer seeks for a boot disk
or floppy.

6.5 Booting with a DOS executable (COM) file

If the computer has to be used with more than one operating systems,
for example using the computer as an X-Terminal in addition to the
already installed NT on the harddrive, etherboot has to be used with
the compile-time option -DASK_BOOT. In case hardware-conflicts between
Windows NT and the installed EPROM exist, creating DOS Executables
```

```
(e.g. using 'make rtl8139.com') can provide a useful remedy. Those DOS
Executables are comparable in their functionality to .rom images and
can be used as substitutes.

In case an existing DOS-bootsector, stored in BOOTSECT.DOS, cannot be
used, creating one has to be done by formatting and installing a harddrive
using DOS before installing NT (see Win-NT Multiboot-HOWTO). In addition
the DOS system files are needed (IO.SYS, MSDOS.SYS, or KERNEL.SYS
when using FreeDOS) and have to be copied into the directory of the
NT loader. If using Autoexec.bat to start the .COM file is desired,
either the particular COMMAND.COM has to be provided or the etherboot
file needs to be renamed as COMMAND.COM. This file will then be started
instead of the DOS-Shell which is useful for avoiding unwanted user
interaction. Afterwards a line has to be added to BOOT.INI as if DOS
was to be booted:

[boot loader]
timeout=20
default=C:\bootsect.dos #add this line if dhcp/tftp should be default action
[operating systems]
multi(0)disk(0)rdisk(0)partition(2)\WINNT="Windows NT Workstation, Version 4.0"
[VGA-Modus]" /basevideo /sos
C:\bootsect.dos="DHCP/TFTP (Linux diskless via etherboot)" #our boot option
```

And here are some comments by Rapp Informatik Systeme GmbH about cbrom.exe versions:

```
Some more remarks for cbrom..

There are several version numbers of cbrom.exe p.e. 1.x and 2.x.
and there is a cbrom called cbrom6.exe. First cbrom.exe with Version
1.x (newest 1.32) is for Award Bios Version 4.5x  and cbrom6.exe is
for Award Bios Version 6.xx.

So because it seems a lot people become confused and use cbrom 1.x
for the new 6.x Bios Award merged this together to a cbrom.exe  with
Version number 2.x ( newest know 2.04) witch now runs on Award 4.5x and
6.xx Bioses.

Now how to find cbrom.exe. Different 1.x Versions of cbrom.exe
could be found on the net, cbrom6.exe seems to be gone. It seems
that Award/Phoenix do all that cbrom is deleted from servers of board
manufactures. So cbrom.exe Vers 2.04 is not available on the net.
If somebody need this please try to send a demand question to the list -
I hope somebody will mail it to you.
```

2. How do I enable the ROM socket on my network adapter? There are no jumpers on the card.

These jumperless cards need a card-specific utility program to enable the ROM. Normally the manufacturer supplies it on a diskette or CDROM. You lost the diskette? If you know the manufacturer, you might be able to get the program from their website. You have a mystery card? Well the first thing to do is to identify the card. If it is an ISA card and made in Taiwan or China it's

almost certainly a NE2000 clone. For some information, try here
(http://www.geocities.com/ken_yap_aus/). If it's a PCI card, then either the BIOS or the Linux PCI
Utilities (http://atrey.karlin.mff.cuni.cz/~mj/pciutils.shtml) should be able to tell you the
manufacturer and device IDs, which you can then look up to convert to names.

3. I would like to boot my laptop diskless from a floppy containing Etherboot.

    The problem is that laptops these days use PCMCIA network adapter cards. These in turn connect to
    the PCMCIA controller when docked. To be able to communicate with the PCMCIA card, Etherboot
    would first have to talk to the PCMCIA controller. Until somebody writes the code to do this...
    Booting from disk is different because the kernel will load the PCMCIA controller code from disk
    first. You could always put a Linux kernel on the boot floppy.

9.10. Drivers

1. There is no Etherboot driver for my network adapter. Can you write me one?

    If I were independently wealthy and had nothing else to do in life, sure! But unfortunately I have a
    day job and Etherboot is a hobby. A couple of the drivers were written for pay and the others were
    written by volunteers. Perhaps you might like to volunteer? If you have a good grasp of C, and
    understand basic hardware concepts, it is quite doable, and not nearly as difficult as writing a Linux
    device driver. See the developer manual (../devman/t1.html). You will have the reward of
    understanding hardware intimately and seeing your work benefit users worldwide.

    If you are a commercial entity, you might consider assigning staff or hiring a volunteer to write the
    driver. You get the benefit of the rest of the Etherboot code infrastructure and users worldwide get to
    appreciate your contribution. Bear in mind license conditions detailed in Section 2.2, of course. NIC
    manufacturers note, this may be one way to attract users to your hardware products. NIC users note,
    petition your NIC manufacturer to support Etherboot.

2. I see that my network adapter is supported in Linux but not in Etherboot. Can I use the Linux driver
    in Etherboot? Or maybe you can adapt the Linux source for me. I can send you the Linux driver
    source if you just say the word.

    No, the structure of Linux and Etherboot drivers are rather different. There are several reasons:
    Linux drivers are more complicated and written to give good performance, whereas Etherboot
    drivers are written to be simple. Linux drivers are interrupt driven, whereas Etherboot drivers are
    polling. Linux drivers have an elaborate support structure, whereas Etherboot drivers are fairly
    self-standing.

    But... you can use Linux drivers as a source of reverse-engineering information. Several of the
    drivers in Etherboot were adapted from Linux drivers. But don't send me the driver source; see
    previous FAQ about volunteering. And I have the latest Linux source anyway, doesn't everyone?

9.11. Miscellaneous

1. I don't understand something, or I have a question not covered by this list.

   Please see Section 2.4 earlier in this document.

10. Acknowledgements

The following people have contributed substantially to Etherboot. If you feel your name has been left out, just let me know and I will fix it up.

Markus Gutschke

   Co-author of Etherboot. He was the person who ported the Netboot suite from FreeBSD. He has enhanced Etherboot with many features, one new driver and has contributed various utilities and addons.

Gero Kuhlmann

   The original mknbi utilities used by Etherboot are from Netboot. He has also clarified the original specification by Jamie Honan.

Jamie Honan

   Jamie started Netboot off by writing the first version that used code from a packet driver.

Martin Renters et. al

   The original authors of Netboot on FreeBSD.

Bruce Evans

   Created bcc compiler used by Etherboot/16 (16 bit ROMs are no longer supported).

Rob de Bath

   Current maintainer of bcc and associated tools like as86, used in older versions of Etherboot to assemble the x86 assembly language files.

Gerd Knorr

   Contributed MASQ for making a boot floppy without DOS.

Adam Richter

   Contributed comboot for making a boot floppy without DOS.

Claus-Justus Heine

   Contributed patch for serial console and NFS swapping. See the contrib/nfs-swap directory for his Web page.

Dickon Reed

Contributed display of loading status and a hack for the 3c509 card.

David Munro

Contributed PCI detection code originally from Linux sources.

Charlie Brady

Donated NE2100 card so that a driver could be written, and helped test the LancePCI driver. Spotted bug with 4.1 header code.

Rogier Wolff

Created Intel EtherExpressPro 100 driver and binary to hex converter.

Vlag Lungu

Contributed patches to work with DHCP. Also contributed a fix to match the received XID against the transmitted one, important in a network with many requesters.

William Arbaugh

Patches for eepro to work with 3.2.

Jean Marc Lacroix

Contributed an improved bin2intelhex.

Jim Hague

Contributed fixes to 3c503 driver for PIO mode, fix to makerom for presetting EPROM bytes, and various endian fixes.

Andrew Coulthurst

Contributed patch for making Intel eepro work in 4.0.

Doug Ambrisko

Contributed patches to start32.S from FreeBSD version to make it boot Windoze after answering N to Boot from Network question. Contributed FreeBSD support and improved serial console support which is now merged into distribution since version 4.2.8. Minor patches to 4.7.21 to make compilation under FreeBSD easier.

Alex Harin

Contributed patches for prepended loaders and makerom to make bootrom PnP and PCI compatible.

Peter Dobcsanyi

Contributed vendor and device IDs for the Netvin NE2000/PCI clone.

adam AT mudlist PERIOD eorbit PERIOD net

Contributed RARP code as alternative to BOOTP/DHCP. Activated by RARP_NOT_BOOTP define.

Daniel Engstrom

Contributed a SMC9000 driver.

Didier Poirot

Contributed an Etherpower II (EPIC 100) driver.

Martin Atkins

Contributed mntnbi for mounting DOS NBIs.

Attila Bogar

Contributed a bug fix to the bootmenu code and a patch to main.c to remove looping menus on failure. Also code for ARP replies and TFTP retransmit (#ifdef CONGESTED). Cleanup of tftp and tftpd.

Nathan R. Neulinger

Found bug due to tu_block being declared signed short in arpa/tftp.h on many platforms when it should be unsigned short.

David Sharp

Contributed a FreeBSD driver for Tulip based cards. Ken Yap ported it to Etherboot. Not tested because code needs to be written for all the variants of the Tulip and also because no hardware available to me.

Greg Beeley

Contributed a 3c905b driver. Be sure to read the release notes in 3c905b.txt before using.

Alex Nemirovsky

Contributed patches to use BIOS call to size memory otherwise Etherboot was trampling on top of 640kB area, which is used by some extended BIOSes. Also contributed patches to pci.c to implement PCI bus support on BIOSes that do not implement BIOS32, or incorrectly.

Günter Knauf

Suggested making the ASK_BOOT prompts more generic and clearer. Also contributed a DOS utility for extracting the identifier string and PCI IDs, if any, out of the boot ROM. Contributed a wake on LAN CGI script.

Klaus Espenlaub

Contributed various cleanup patches to the code especially in the bootmenu area, fixes for the NE2000 driver, as well as a completely revamped start32.S. Also introduced Rainer Bawidamann's code, see next paragraph. Contributed further improvements in Realtek 8139 driver. Did a major rewrite from 4.4.5 to 4.5.5, see doc/maint/LOG.

Rainer Bawidamann

Contributed a Realtek 8139 driver.

Georg Baum

contributed a Schneider & Koch G16 driver.

jluke AT deakin PERIOD edu PERIOD au

sent in a fix for the WD/SMC8013 which I finally verified.

Mark Burazin

contributed a fix for Compex RL2000 NICs.

Matthias Meixner

found a receive status bug in the RTL8139 driver.

Jim McQuillan

provided changes to support the SMC1211 which uses the RTL8139 chip.

Steve Smith

Extended the 3c905b driver for other members of the 90x family. Be sure to read the release notes in 3c90x.txt before using. Modified loader.S for some BIOSes that don't behave correctly with INT19H.

John Finlay

Wrote a utility for programming EEPROMs on 3c90x in situ.

Nick Lopez

Contributed change to tulip.c to handle Macronix 98715 (Tulip clone).

Matt Hortman

Contributed fix to eepro100 driver that fixes incorrect latency setting. Also Makefile rule for .lzfd0.

Marty Connor

Contributed new Tulip driver ntulip.c. Reduced RTL8139 footprint. Added support for Netgear FA310TX (Tulip clone, LC82C168 chip). Support for 3Com905C. Romutil for 905C, which have block erase EEPROMs. Contributed the development of liloprefix.S through thinguin.org. Finally made the ROM images conformant PnP boot ROM images. Wrote the SiS900 driver. Made Tulip driver work for many more variants. Wrote the National Semiconductor DP83815 driver, under funding from Sicom System (http://www.sicompos.com/).

Anders Larsen

contributed mkQNXnbi, for generating boot images from QNX kernels.

Bernd Wiebelt

contributed code to request vendor tags in DHCP.

Paolo Marini

contributed the Via-Rhine driver.

Adam Fritzler

contributed 3c529 (MCA version of 3c509) support in driver.

Shusuke Nisiyama

contributed a 3c595 (may work for 3c590) driver.

Igor V. Kovalenko

contributed a Winbind W89C840 driver.

Gary Byers

of thinguin.org wrote the LILO prefix program liloprefix.S.

Donald Christensen

converted all the as86/nasm .S files to gas format, thus allowing Etherboot to be built without requiring as86/nasm.

Luigi Rizzo

contributed a bootloader from FreeBSD that works for both floppy and hard disk partitions. This obsoletes floppyload.S. Patch to do adaptive timeout in NFS booting.

Michael Sinz

contributed patches for loading debug symbols when booting FreeBSD.

Jean-Jacques Michel

contributed patches to the via-rhine.c driver to make it work for the VT6102 model as used on some DFE530-TX Rev.A3 NICs.

Peter Kögel

contributed patches to the SiS900 driver to make it work for the SiS630e/SiS730s.

Eric Biederman

improved the recovery logic of main.c and lots of other changes. Contributed E820H memory sizing logic. Reworked PCI logic. Made more compatible with LinuxBIOS. Lots of small fixes. Reworked the architecture completely to be much more portable. Wrote a lot of new code. A list of his contributions would take several paragraphs.

Peter Lister and Vasil Vasilev

contributed changes to generate .pxe image that can be booted using a PXE booter.

Fred Gray

contributed changes in tulip.c to check for a duplex connection and to modify the controller register if so.

Mark G

of Inprimis Technologies contributed another FA311 (National Semiconductor DP83815) driver, also based on the Donald Becker Linux driver.

Armin Schindler

contributed a patch to allow booting LynxOS KDI images.

Christopher Li

contributed an Intel E1000 gigabit Ethernet driver.

Rohit Jalan

contributed a patch for FreeBSD style PXE booting.

Andrew Bettison

sent in a patch to run the SMC EtherEZ in PIO mode, required for some motherboards.

Michael Brown

contributed the first wireless NIC drivers for the Prism chipset.

Timothy Legge

contributed a 3c515 driver, ISA PnP implementation, sundance, tlan, and pcnet32 drivers, enabled multicast in 3c509, 3c515, 3c595, pcnet32, rtl8139, sundance, tlan, and via-rhine drivers, and contributed new multicast tftp implementation (proto_tftm).

Cai Qiang

fixed the WinCE loader.

11. Source copyrights

The boot code from FreeBSD is under the BSD license. The code taken from the Linux PCI subsystem and Linux NIC drivers are under GPL. Some source files have been put under GPL by their authors. Hence the Etherboot distribution is in general under the GPL, but you may use parts of it derived from FreeBSD under FreeBSD rules. Simply speaking, the GPL says that if you distribute a binary derived from Etherboot code (this includes boot ROMs) you have to provide, or promise to provide on demand, the source code. The full conditions of the GPL are specified in the file COPYING.

Here are the copyright details of the source, file by file:

```
Unless specifically noted, a file is under the GPL.  GPLed files are in
general either from Linux or have been explicitly put under GPL by the
authors.  A few files are inherited from FreeBSD netboot and therefore
can be used under BSD or GPL.


File    Copyright status

core/misc.c   BSD
drivers/net/3c509.c  BSD
drivers/net/3c509.h  BSD
drivers/net/3c595.c  BSD
drivers/net/3c595.h  BSD
```

```
drivers/net/3c90x.c  Open Source
drivers/net/epic100.c  None
drivers/net/epic100.h  None
drivers/net/ns8390.c  BSD
drivers/net/ns8390.h  BSD
drivers/net/tulip.c  BSD
arch/i386/include/bits/string.h None
util/lzhuf.c   Open Source
```

A. List of supported NICs

The following is the current NIC configuration file as of 2003-09-23. .

```
# This is an automatically generated file, do not edit
# It does not affect anything in the build, it's only for rom-o-matic

family drivers/net/skel
skel-pci 0x0000,0x0000 Skeleton PCI Adaptor
skel-isa - Skeleton ISA driver

family drivers/net/3c595
3c590 0x10b7,0x5900 3Com590
3c595 0x10b7,0x5950 3Com595
3c595-1 0x10b7,0x5951 3Com595
3c595-2 0x10b7,0x5952 3Com595
3c900-tpo 0x10b7,0x9000 3Com900-TPO
3c900-t4 0x10b7,0x9001 3Com900-Combo
3c900b-tpo 0x10b7,0x9004 3Com900B-TPO
3c900b-combo 0x10b7,0x9005 3Com900B-Combo
3c900b-tpb2 0x10b7,0x9006 3Com900B-2/T
3c900b-fl 0x10b7,0x900a 3Com900B-FL
3c980-cyclone-1 0x10b7,0x9800 3Com980-Cyclone
3c9805-1 0x10b7,0x9805 3Com9805
3csoho100-tx-1 0x10b7,0x7646 3CSOHO100-TX
3c450-1 0x10b7,0x4500 3Com450 HomePNA Tornado

family drivers/net/3c90x
3c905-tpo 0x10b7,0x9000 3Com900-TPO
3c905-t4 0x10b7,0x9001 3Com900-Combo
3c905-tpo100 0x10b7,0x9050 3Com905-TX
3c905-combo 0x10b7,0x9051 3Com905-T4
3c905b-tpo 0x10b7,0x9004 3Com900B-TPO
3c905b-combo 0x10b7,0x9005 3Com900B-Combo
3c905b-tpb2 0x10b7,0x9006 3Com900B-2/T
3c905b-fl 0x10b7,0x900a 3Com900B-FL
3c905b-tpo100 0x10b7,0x9055 3Com905B-TX
```

```
3c905b-t4 0x10b7,0x9056 3Com905B-T4
3c905b-9058 0x10b7,0x9058 3Com905B-9058
3c905b-fx 0x10b7,0x905a 3Com905B-FL
3c905c-tpo 0x10b7,0x9200 3Com905C-TXM
3c980 0x10b7,0x9800 3Com980-Cyclone
3c9805 0x10b7,0x9805 3Com9805
3csoho100-tx 0x10b7,0x7646 3CSOHO100-TX
3c450 0x10b7,0x4500 3Com450 HomePNA Tornado


family drivers/net/eepro100
id1029 0x8086,0x1029 Intel EtherExpressPro100 ID1029
id1030 0x8086,0x1030 Intel EtherExpressPro100 ID1030
82801cam 0x8086,0x1031 Intel 82801CAM (ICH3) Chipset Ethernet Controller
eepro100-1032 0x8086,0x1032 Intel PRO/100 VE Network Connection
eepro100-1033 0x8086,0x1033 Intel PRO/100 VM Network Connection
eepro100-1034 0x8086,0x1034 Intel PRO/100 VM Network Connection
eepro100-1035 0x8086,0x1035 Intel 82801CAM (ICH3) Chipset Ethernet Controller
eepro100-1036 0x8086,0x1036 Intel 82801CAM (ICH3) Chipset Ethernet Controller
eepro100-1037 0x8086,0x1037 Intel 82801CAM (ICH3) Chipset Ethernet Controller
id1038 0x8086,0x1038 Intel PRO/100 VM Network Connection
82562et 0x8086,0x1039 Intel PRO100 VE 82562ET
id103a 0x8086,0x103a Intel Corporation 82559 InBusiness 10/100
82562etb 0x8086,0x103b Intel PRO100 VE 82562ETB
eepro100-103c 0x8086,0x103c Intel PRO/100 VM Network Connection
eepro100-103d 0x8086,0x103d Intel PRO/100 VE Network Connection
eepro100-103e 0x8086,0x103e Intel PRO/100 VM Network Connection
82551qm 0x8086,0x1059 Intel PRO/100 M Mobile Connection
82559er 0x8086,0x1209 Intel EtherExpressPro100 82559ER
82865 0x8086,0x1227 Intel 82865 EtherExpress PRO/100A
82556 0x8086,0x1228 Intel 82556 EtherExpress PRO/100 Smart
eepro100 0x8086,0x1229 Intel EtherExpressPro100
82562em 0x8086,0x2449 Intel EtherExpressPro100 82562EM
82562-1 0x8086,0x2459 Intel 82562 based Fast Ethernet Connection
82562-2 0x8086,0x245d Intel 82562 based Fast Ethernet Connection
82562ez 0x8086,0x1050 Intel 82562EZ Network Connection
eepro100-5200 0x8086,0x5200 Intel EtherExpress PRO/100 Intelligent Server
eepro100-5201 0x8086,0x5201 Intel EtherExpress PRO/100 Intelligent Server


family drivers/net/e1000
e1000-82542 0x8086,0x1000 Intel EtherExpressPro1000
e1000-82543gc-fiber 0x8086,0x1001 Intel EtherExpressPro1000 82543GC Fiber
e1000-82543gc-copper 0x8086,0x1004 Intel EtherExpressPro1000 82543GC Copper
e1000-82544ei-copper 0x8086,0x1008 Intel EtherExpressPro1000 82544EI Copper
e1000-82544ei-fiber 0x8086,0x1009 Intel EtherExpressPro1000 82544EI Fiber
e1000-82544gc-copper 0x8086,0x100c Intel EtherExpressPro1000 82544GC Copper
e1000-82544gc-lom 0x8086,0x100d Intel EtherExpressPro1000 82544GC LOM
e1000-82540em 0x8086,0x100e Intel EtherExpressPro1000 82540EM
e1000-82545em-copper 0x8086,0x100f Intel EtherExpressPro1000 82545EM Copper
e1000-82546eb-copper 0x8086,0x1010 Intel EtherExpressPro1000 82546EB Copper
e1000-82545em-fiber 0x8086,0x1011 Intel EtherExpressPro1000 82545EM Fiber
e1000-82546eb-fiber 0x8086,0x1012 Intel EtherExpressPro1000 82546EB Copper
e1000-82541ei 0x8086,0x1013 Intel EtherExpressPro1000 82541EI
e1000-82540em-lom 0x8086,0x1015 Intel EtherExpressPro1000 82540EM LOM
```

```
e1000-82540ep-lom 0x8086,0x1016 Intel EtherExpressPro1000 82540EP LOM
e1000-82540ep 0x8086,0x1017 Intel EtherExpressPro1000 82540EP
e1000-82541ep 0x8086,0x1018 Intel EtherExpressPro1000 82541EP
e1000-82547ei 0x8086,0x1019 Intel EtherExpressPro1000 82547EI
e1000-82546eb-quad-copper 0x8086,0x101d Intel EtherExpressPro1000 82546EB Quad Copper
e1000-82540ep-lp 0x8086,0x101e Intel EtherExpressPro1000 82540EP LP


family drivers/net/tg3
tg3-5700 0x14e4,0x1644 Broadcom Tigon 3 5700
tg3-5701 0x14e4,0x1645 Broadcom Tigon 3 5701
tg3-5702 0x14e4,0x1646 Broadcom Tigon 3 5702
tg3-5703 0x14e4,0x1647 Broadcom Tigon 3 5703
tg3-5704 0x14e4,0x1648 Broadcom Tigon 3 5704
tg3-5702FE 0x14e4,0x164d Broadcom Tigon 3 5702FE
tg3-5705 0x14e4,0x1653 Broadcom Tigon 3 5705
tg3-5705_2 0x14e4,0x1654 Broadcom Tigon 3 5705_2
tg3-5705M 0x14e4,0x165d Broadcom Tigon 3 5705M
tg3-5705M_2 0x14e4,0x165e Broadcom Tigon 3 5705M_2
tg3-5782 0x14e4,0x1696 Broadcom Tigon 3 5782
tg3-5788 0x14e4,0x169c Broadcom Tigon 3 5788
tg3-5702X 0x14e4,0x16a6 Broadcom Tigon 3 5702X
tg3-5703X 0x14e4,0x16a7 Broadcom Tigon 3 5703X
tg3-5704S 0x14e4,0x16a8 Broadcom Tigon 3 5704S
tg3-5702A3 0x14e4,0x16c6 Broadcom Tigon 3 5702A3
tg3-5703A3 0x14e4,0x16c7 Broadcom Tigon 3 5703A3
tg3-5901 0x14e4,0x170d Broadcom Tigon 3 5901
tg3-5901_2 0x14e4,0x170e Broadcom Tigon 3 5901_2
tg3-9DXX 0x1148,0x4400 Sysconnet 9DXX
tg3-9MXX 0x1148,0x4500 Sysconnet 9MXX
tg3-ac1000 0x173b,0x03e8 Altima AC1000
tg3-ac1001 0x173b,0x03e9 Altima AC1001
tg3-ac9100 0x173b,0x03ea Altima AC9100
tg3-ac1003 0x173b,0x03eb Altima AC1003


family drivers/net/pcnet32
lancepci 0x1022,0x2000 AMD Lance/PCI
pcnetfastiii 0x1022,0x2625 AMD Lance/PCI PCNet/32
amdhomepna 0x1022,0x2001 AMD Lance/HomePNA


family drivers/net/tulip
dc21040 0x1011,0x0002 Digital Tulip
ds21140 0x1011,0x0009 Digital Tulip Fast
dc21041 0x1011,0x0014 Digital Tulip+
ds21142 0x1011,0x0019 Digital Tulip 21142
mx98713 0x10d9,0x0512 Macronix MX987x3
mx98715 0x10d9,0x0531 Macronix MX987x5
mxic-98715 0x1113,0x1217 Macronix MX987x5
lc82c115 0x11ad,0xc115 LinkSys LNE100TX
82c168 0x11ad,0x0002 Netgear FA310TX
dm9100 0x1282,0x9100 Davicom 9100
dm9102 0x1282,0x9102 Davicom 9102
dm9009 0x1282,0x9009 Davicom 9009
centaur-p 0x1317,0x0985 ADMtek Centaur-P
```

```
an981 0x1317,0x0981 ADMtek AN981 Comet
an983 0x1113,0x1216 ADMTek AN983 Comet
an983b 0x1317,0x9511 ADMTek Comet 983b
centaur-c 0x1317,0x1985 ADMTek Centaur-C
intel21145 0x8086,0x0039 Intel Tulip
ax88140 0x125b,0x1400 ASIX AX88140
rl100tx 0x11f6,0x9881 Compex RL100-TX
xircomtulip 0x115d,0x0003 Xircom Tulip
tulip-0981 0x104a,0x0981 Tulip 0x104a 0x0981
tulip-2774 0x104a,0x2774 Tulip 0x104a 0x2774
tulip-9511 0x1113,0x9511 Tulip 0x1113 0x9511
tulip-1561 0x1186,0x1561 Tulip 0x1186 0x1561
tulip-a120 0x1259,0xa120 Tulip 0x1259 0xa120
tulip-ab02 0x13d1,0xab02 Tulip 0x13d1 0xab02
tulip-ab03 0x13d1,0xab03 Tulip 0x13d1 0xab03
tulip-ab08 0x13d1,0xab08 Tulip 0x13d1 0xab08
lanfinity 0x14f1,0x1803 Conexant LANfinity
tulip-8410 0x1626,0x8410 Tulip 0x1626 0x8410
tulip-ab09 0x1737,0xab09 Tulip 0x1737 0xab09

family drivers/net/davicom
davicom9100 0x1282,0x9100 Davicom 9100
davicom9102 0x1282,0x9102 Davicom 9102
davicom9009 0x1282,0x9009 Davicom 9009
davicom9132 0x1282,0x9132 Davicom 9132

family drivers/net/rtl8139
rtl8129 0x10ec,0x8129 Realtek 8129
rtl8139 0x10ec,0x8139 Realtek 8139
rtl8139b 0x10ec,0x8138 Realtek 8139B
dfe538 0x1186,0x1300 DFE530TX+/DFE538TX
smc1211-1 0x1113,0x1211 SMC EZ10/100
smc1211 0x1112,0x1211 SMC EZ10/100
delta8139 0x1500,0x1360 Delta Electronics 8139
addtron8139 0x4033,0x1360 Addtron Technology 8139
dfe690txd 0x1186,0x1340 D-Link DFE690TXD
fe2000vx 0x13d1,0xab06 AboCom FE2000VX
allied8139 0x1259,0xa117 Allied Telesyn 8139
fnw3603tx 0x14ea,0xab06 Planex FNW-3603-TX
fnw3800tx 0x14ea,0xab07 Planex FNW-3800-TX
clone-rtl8139 0xffff,0x8139 Cloned 8139

family drivers/net/via-rhine
dlink-530tx 0x1106,0x3065 VIA 6102
via-rhine-6105 0x1106,0x3106 VIA 6105
dlink-530tx-old 0x1106,0x3043 VIA 3043
via6105m 0x1106,0x3053 VIA 6105M
via-rhine-old 0x1106,0x6100 VIA 86C100A

family drivers/net/w89c840
winbond840 0x1050,0x0840 Winbond W89C840F
compexrl100atx 0x11f6,0x2011 Compex RL100ATX
```

```
family drivers/net/sis900
sis900 0x1039,0x0900 SIS900
sis7016 0x1039,0x7016 SIS7016

family drivers/net/natsemi
dp83815 0x100b,0x0020 DP83815

family drivers/net/fa311
fa311too 0x100b,0x0020 DP83815

family drivers/net/prism2_plx
ma301 0x1385,0x4100 Netgear MA301
3c-airconnect 0x10b7,0x7770 3Com AirConnect
ss1023 0x111a,0x1023 Siemens SpeedStream SS1023
correga 0x15e8,0x0130 Correga
smc2602w 0x1638,0x1100 SMC EZConnect SMC2602W
gl24110p 0x16ab,0x1100 Global Sun Tech GL24110P
16ab-1101 0x16ab,0x1101 Unknown
wdt11 0x16ab,0x1102 Linksys WDT11
usr2415 0x16ec,0x3685 USR 2415
f5d6000 0xec80,0xec00 Belkin F5D6000
emobility 0x126c,0x8030 Nortel emobility

family drivers/net/prism2_pci
prism2_pci 0x1260,0x3873 Harris Semiconductor Prism2.5 clone
hwp01170 0x1260,0x3873 ActionTec HWP01170
dwl520 0x1260,0x3873 DLink DWL-520

family drivers/net/ns8390
rtl8029 0x10ec,0x8029 Realtek 8029
dlink-528 0x1186,0x0300 D-Link DE-528
winbond940 0x1050,0x0940 Winbond NE2000-PCI
winbond940f 0x1050,0x5a5a Winbond W89c940F
compexrl2000 0x11f6,0x1401 Compex ReadyLink 2000
ktiet32p2 0x8e2e,0x3000 KTI ET32P2
nv5000sc 0x4a14,0x5000 NetVin NV5000SC
holtek80232 0x12c3,0x0058 Holtek HT80232
holtek80229 0x12c3,0x5598 Holtek HT80229
surecom-ne34 0x10bd,0x0e34 Surecom NE34
via86c926 0x1106,0x0926 Via 86c926
wd - WD8003/8013, SMC8216/8416, SMC 83c790 (EtherEZ)
ne - NE1000/2000 and clones
3c503 - 3Com503, Etherlink II[/16]

family drivers/net/epic100
epic100 0x10b8,0x0005 SMC EtherPowerII
smc-83c175 0x10b8,0x0006 SMC EPIC/C 83c175

family drivers/net/3c509
3c509 - 3c509, ISA/EISA
3c529 - 3c529 == MCA 3c509

family drivers/net/3c515
```

```
3c515 - 3c515, Fast EtherLink ISA

family drivers/net/eepro
eepro - Intel Etherexpress Pro/10

family drivers/net/cs89x0
cs89x0 - Crystal Semiconductor CS89x0

family drivers/net/depca
depca - Digital DE100 and DE200

family drivers/net/sk_g16
sk_g16 - Schneider and Koch G16

family drivers/net/smc9000
smc9000 - SMC9000

family drivers/net/sundance
sundance 0x13f0,0x0201 ST201 Sundance 'Alta' based Adaptor
dfe530txs 0x1186,0x1002 D-Link DFE530TXS (Sundance ST201 Alta)

family drivers/net/tlan
netel10 0x0e11,0xae34 Compaq Netelligent 10 T PCI UTP
netel100 0x0e11,0xae32 Compaq Netelligent 10/100 TX PCI UTP
netflex3i 0x0e11,0xae35 Compaq Integrated NetFlex-3/P
thunder 0x0e11,0xf130 Compaq NetFlex-3/P
netflex3b 0x0e11,0xf150 Compaq NetFlex-3/P
netel100pi 0x0e11,0xae43 Compaq Netelligent Integrated 10/100 TX UTP
netel100d 0x0e11,0xae40 Compaq Netelligent Dual 10/100 TX PCI UTP
netel100i 0x0e11,0xb011 Compaq Netelligent 10/100 TX Embedded UTP
oc2183 0x108d,0x0013 Olicom OC-2183/2185
oc2325 0x108d,0x0012 Olicom OC-2325
oc2326 0x108d,0x0014 Olicom OC-2326
netelligent_10_100_ws_5100 0x0e11,0xb030 Compaq Netelligent 10/100 TX UTP
netelligent_10_t2 0x0e11,0xb012 Compaq Netelligent 10 T/2 PCI UTP/Coax

family drivers/disk/ide_disk
ide_disk 0x0000,0x0000 Generic IDE disk support

family drivers/disk/pc_floppy
pc_floppy - Generic PC Floppy support
```

Even if your NIC does not appear in the list, it may still be supported if the chip is one of those supported. Many OEMs use chips from foundries. An exhaustive list of brand names is impossible. The best strategy is to read the number on the LAN controller chip on the board. The following drivers have notes in the src/drivers/net directory, please read them:

```
3c515.txt
3c90x.txt
```

```
cs89x0.txt
sis900.txt
tulip.txt
```

B. Build options available

```
User interaction options:

-DASK_BOOT=n
  Ask "Boot from (N)etwork ... or (Q)uit? "
  at startup, timeout after n seconds (0 = no timeout).
  If unset, boot immediately using the default.
-DBOOT_FIRST
-DBOOT_SECOND
-DBOOT_THIRD
  On timeout or Return key from previous
  question, selects the order to try to boot from
  various devices.
  (alternatives: BOOT_NIC, BOOT_DISK,
   BOOT_FLOPPY, BOOT_NOTHING)
  See etherboot.h for prompt and answer strings.
-DBOOT_INDEX The device to boot from 0 == any device.
  1 == The first nic found.
  2 == The second nic found
  ...
  BOOT_INDEX only applies to the BOOT_FIRST.  BOOT_SECOND
  and BOOT_THIRD search through all of the boot devices.
-DBAR_PROGRESS
  Use rotating bar instead of sequential dots
  to indicate an IP packet transmitted.

Boot order options:

-DBOOT_CLASS_FIRST
-DBOOT_CLASS_SECOND
-DBOOT_CLASS_THIRD
  Select the priority of the boot classes
  Valid values are:
   BOOT_NIC
   BOOT_DISK
   BOOT_FLOPPY

Boot autoconfiguration protocol options:
```

```
-DNO_DHCP_SUPPORT
  Use BOOTP instead of DHCP.
-DRARP_NOT_BOOTP
  Use RARP instead of BOOTP/DHCP.
-DREQUIRE_VCI_ETHERBOOT
  Require an encapsulated Vendor Class Identifier
  of "Etherboot" in the DHCP reply
  Requires DHCP support.
-DALLOW_ONLY_ENCAPSULATED
  Ignore Etherboot-specific options that are not within
  the Etherboot encapsulated options field.  This option
  should be enabled unless you have a legacy DHCP server
  configuration from the bad old days before the use of
  encapsulated Etherboot options.
-DDEFAULT_BOOTFILE="default_bootfile_name"
  Define a default bootfile for the case where your DHCP
  server does not provide the information.  Example:
    -DDEFAULT_BOOTFILE="tftp:///tftpboot/kernel"
  If you do not specify this option, then DHCP offers that
  do not specify bootfiles will be ignored.


NIC tuning parameters:


-DALLMULTI
  Turns on multicast reception in the NICs.


Boot tuning parameters:


-DCONGESTED
  Turns on packet retransmission.  Use it on a
  congested network, where the normal operation
  can't boot the image.
-DBACKOFF_LIMIT
  Sets the maximum RFC951 backoff exponent to n.
  Do not set this unreasonably low, because on networks
  with many machines they can saturate the link
  (the delay corresponding to the exponent is a random
  time in the range 0..3.5*2^n seconds). Use 5 for a
  VERY small network (max. 2 minutes delay), 7 for a
  medium sized network (max. 7.5 minutes delay) or 10
  for a really huge network with many clients, frequent
  congestions (max. 1  hour delay).  On average the
  delay time will be half the maximum value.  If in
  doubt about the consequences, use a larger value.
  Also keep in mind that the number of retransmissions
  is not changed by this setting, so the default of 20
  may no longer be appropriate.  You might need to set
  MAX_ARP_RETRIES, MAX_BOOTP_RETRIES, MAX_TFTP_RETRIES
  and MAX_RPC_RETRIES to a larger value.
-DTIMEOUT=n
  Use with care!! See above.
  Sets the base of RFC2131 sleep interval to n.
  This can be used with -DBACKOFF_LIMIT=0 to get a small
```

and constant (predictable) retry interval for embedded
devices. This is to achieve short boot delays if both
the DHCP Server and the embedded device will be powered
on the same time. Otherwise if the DHCP server is ready
the client could sleep the next exponentially timeout,
e.g. 70 seconds or more. This is not what you want.
n should be a multiple of TICKS_PER_SEC (18).

Boot device options:

-DCAN_BOOT_DISK
  Can boot from floppy/hd if bootimage matches the
  pattern "/dev/fhsd*".
-DTRY_FLOPPY_FIRST
  If > 0, tries that many times to read the boot
  sector from a floppy drive before booting from
  ROM. If successful, does a local boot.
  It assumes the floppy is bootable.
  Requires -DCAN_BOOT_DISK.
-DEMERGENCYDISKBOOT
  If no BOOTP server can be found, then boot from
  local disk. The accessibility of the TFTP server
  has no effect, though! So configure your BOOTP
  server properly. You should probably reduce
  MAX_BOOTP_RETRIES to a small number like 3.

Boot image options:

-DTAGGED_IMAGE
  Add tagged image kernel boot support (recommended).
-DAOUT_IMAGE
  Add a.out kernel boot support (generic).
-DELF_IMAGE
  Add generic ELF kernel boot support (recommended).
-DEL64F_IMAGE
  Add generic ELF64 kernel boot support (useful for > 4GB disks).
-DWINCE_IMAGE
  Add the ability to boot WINCE.... now only sis630 OK!
-DFREEBSD_PXEEMU
  Add the ability to boot PXE images... only FreeBSD supported
-DX86_BOOTSECTOR_IMAGE
  Add the ability to boot 512 byte x86 boot sectors
-DIMAGE_MULTIBOOT
  Add Multiboot image support (currently only
  for ELF images).
  Without this, generic ELF support is selected.
-DIMAGE_FREEBSD
  Add FreeBSD image loading support (requires at least
  -DAOUT_IMAGE and/or -DELF_IMAGE).
-DFREEBSD_KERNEL_ENV
  Pass in FreeBSD kernel environment
-DAOUT_LYNX_KDI
  Add Lynx a.out KDI support

```
-DMULTICAST_LEVEL1
  Support for sending multicast packets
-DMULTICAST_LEVEL2
  Support for receiving multicast packets
-DDOWNLOAD_PROTO_TFTP
  If defined, boots by TFTP (recommended).
-DDOWNLOAD_PROTO_NFS
  If defined, boots from a NFS mount and disables
  TFTP loading. Default is DOWNLOAD_PROTO_TFTP
  if neither is defined.
-DDOWNLOAD_PROTO_SLAM
  If defined, boots via Scalable Local Area Multicast.
-DDOWNLOAD_PROTO_TFTM
  If defined, enables booting via TFTP Multicast mode.
-DSAFEBOOTMODE
  Enables "Safe Boot Mode": Only boot images after verification
  with public-key-cryptography, this is WORK IN PROGRESS
  must be or'ed from one publickey storage method and
  one nbi-digest method value
  0  = store key in code (safeboot_key.h)
  1  = store key in ROM somewhere else (to do)
  0 = crypted digest in first 512 bytes -
      as bytes 446...509 (compatible with most NBIs!?)
  16 = crypted digest as ELF block (to do... Eric?)
  So for now, only valid mode is "0"


Console options:


-DCONSOLE_FIRMWARE
  Set for firmware/BIOS provided (default if nothing else is set).
  Normally this is shows up on your CRT.
-DCONSOLE_SERIAL
  Set for serial console.
-DCONSOLE_DUAL
  Both of the above
-DCOMCONSOLE
  Set port, e.g. 0x3F8.
-DCONSPEED
  Set speed, e.g. 57600.
-DCOMPARM
  Set Line Control Register value for data bits, stop
  bits and parity. See a National Semiconditor 8250/
  16450/16550 data sheet for bit meanings.
  If undefined, defaults to 0x03 = 8N1.
-DCOMPRESERVE
  Ignore COMSPEED and COMPARAM and instead preserve
  the com port parameters from the previous user
  of the com port.  Examples of previous user are a BIOS
  that implements console redirection, lilo and LinuxBIOS.
  This makes it trivial to keep the serial port
  speed setting in sync between multiple users.
  You set the speed in the first user and the
  rest follow along.
```

```
Obscure options you probably don't need to touch:

-DPOWERSAVE
  Halt the processor when waiting for keyboard input
  which saves power while waiting for user interaction.
  Good for compute clusters and VMware emulation.
  But may not work for all CPUs.
-DRELOCATE
  After starting etherboot relocate to the top
  of memory.  This allows loading fairly arbitrary
  rom images.  This doesn't work with a couple of
  drivers, e.g. lance.

BUS options:

-DCONFIG_PCI
  Include support for devices using the pci bus.
-DCONFIG_ISA
  Include support for devices using isa bus.
```

C. Security, 1 July 1997

Markus Gutschke, gutschk AT math PERIOD uni-muenster PERIOD de

C.1. Introduction

This documentation has been written and is copyrighted 1997 by Markus Gutschke <gutschk AT math PERIOD uni-muenster PERIOD de>. You are free to distribute this file as long as you do not change its contents. I appreciate comments and will consider them in future revisions. If you have any questions, comments, or suggestions, please also send carbon copies of your e-mail message to both Ken Yap <ken_yap AT users PERIOD sourceforge PERIOD net> and Gero Kuhlmann <gero AT gkminix PERIOD han PERIOD de>. I will happily include any of your extensions, but I would like to avoid the proliferation of different incompatible revisions of this document. If these conditions are a problem to you, then feel free to contact me.

C.2. SCOPE OF THIS TEXT

Any computer that is either physically accessible or can remotely be contacted over a network is potentially threatened by attempts to circumvent its security measures. The PC architecture is especially insecure and using a BOOT-Prom can help preventing some of the more obvious attacks. On the other

hand, it also enables some attacks that are not possible against machines that boot from local mass storage devices (floppy, harddisk, ...). As a general rule, you can assume that it is impossible to protect your system from all conceivable attacks, but you can try to minimize the chance of an attack, try to minimize the seriousness of the damage caused and try to help in detecting attacks at the earliest possible moment. This text discusses some of the issues involved and tries to raise awareness of security problems; it cannot provide generic solutions to all of these problems, but should help you in appreciating the need for proper administration and regular security updates.

C.3. PC ARCHITECTURE

The PC architecture was never designed with security considerations in mind. Running an operating system such as DOS or Windows, allows the user full control over the hardware. There are no effective measures for preventing him from modifying data on local mass storage devices, reading confidential data that is stored locally, installing trojan horses and programs that intercept user input, monitoring all data packets that are sent on the local ethernet segment, sending ethernet packages with faked authorization information, and a whole lot of other potentially dangerous actions.

Modern operating systems and suitable hardware extensions make these attacks more difficult, but as long as the user can still launch arbitrary programs that have full hardware access, there is not much protection that can be achieved in this way. Therefore, the most important security measures are those that prevent a malicious user from executing programs from arbitrary external sources.

The PC must be physically protected, so that there is no way of replacing the harddrive (either for reading its contents or for installing compromised software), installing a modified system BIOS (if your system has a Flash BIOS, then make sure that it is always write protected and that write protection actually works!), connecting it to a different network, removing extra hardware that offers security features, or performing some other hardware modification.

If the PC offers exchangeable mass storage devices (floppy, ZIP drive, CD-ROM, ...), the system must be configured to never boot from these devices.

This is not as easy as it might sound. Most motherboards come with a system BIOS which offers back-doors to their password. So configuring the BIOS to never boot from exchangeable devices, will not prevent any determined hacker from modifying this setting. These generically accepted passwords might not be known to your average users, but you can assume that even a poorly informed hacker will know about them. Besides, if your system allows for reading the contents of the CMOS memory chip (e.g. by starting a suitable DOS program such the debugger which ships with DOS), then the password can be computed from this data. While you should still make sure, that your BIOS has password protection, this is only to be considered as a mild deterrent.

A more useful protection is achieved by installing extra firmware, which requires a password before booting from a local device. This firmware should be written in a way which prevents the system BIOS from disabling it. This is the default configuration for the "etherboot" BOOT-Prom, but it might still be circumvented, if your system BIOS allows for disabling certain memory areas from being scanned for ROMs. If this is the case, then complain to your system vendor and replace the machine; it is unsuitable for being used in a publically accessible place. Also, in a security conscious environment, there must be no way of escaping from the control of the BOOT-Prom; this means, that you must not enable the "-DEMERGENCYDISKBOOT" option when compiling the "etherboot" software and you must never offer an empty image name in the image menu (c.f. README.VendorTags). If you want to offer booting

from a local device, then specify the full device name and either enable the "-DFLOPPY" option or store a boot image (as generated by "mknbi-blkdev") under this name on the TFTP server.

It might be a good idea to install non-functional bootcode in the master boot record of your harddisk and install any locally required boot code in the boot sectors of the individual partitions. Both "etherboot" and "mknbi-blkdev" know how to boot from partitions such as "/dev/hda1" (first primary partition) or "/dev/hdb5" (first logical partion on the second harddrive).

If you have the choice, then do not offer operating systems which come without effective password protection and virtualization. This basically rules out all of the following: DOS, Windows, Win95, ...

You should be aware of the fact, that offering one "dangerous" operating system, makes all other operating systems vulnerable as well. A noteworthy example is the availability of tools which allow an arbitrary DOS user to read and modify the contents of a Linux harddisk partition (similar tools are supposedly available for accessing NT partions).

C.4. PASSWORDS PROTECTED BOOT-PROMS

Password protected BOOT-Proms are a considerable improvement over the standard security measures offered by the PC architecture, but they are still vulnerable to a variety of attacks. Most of these attacks are related to the weaknesses of the BOOTP and TFTP protocol and these issues are discussed in another section of text.

As the BOOT-Prom cannot store the passwords locally, it has to request them from the BOOTP/TFTP server. The BOOTP and TFTP protocols do not allow for any elaborate challenge/response authorization scheme. Thus the BOOT-Prom requests the password from the server and subsequently compares it with the password as input by the user. As packets transmitted on the ethernet can easily be monitored, the server sends a MD5 message digest (as invented by Ron Rivest/"RSA Data Security, Inc.") of the actual password. The BOOT-Prom computes the MD5 value for the user input and compares these two values. It is generally believed that there is no way of computing a valid plain text from a known MD5 message digest, other than comparing it against all conceivable input texts. Thus, this approach is still vulnerable against dictionary attacks. You should aim for using long passwords (although, anything beyond 20 characters is unlikely to considerably improve the security of the password) which are not listed in any dictionary. Make sure that these passwords are memorized and not available in un-encrypted form. Replace the passwords regularly and do not offer more menu entries in the BOOTP data than are absolutely necessary.

C.5. BOOTING DOS

If you have to offer DOS or a related operating system, then do not fool yourself into believing that you can install security software in one of its configuration files. All of these mechanisms can easily be avoided. In many cases, even average users can figure out how to do this.

C.6. BOOTING LINUX

There are a variety of ways that allow for booting Linux in single user mode. The most common

techniques involve passing a suitable option on the kernel command line (i.e. "single") or crashing the filesystem by power cycling the machine; this in turn will result in fsck being invoked at the next system start, which will sometimes drop you into single user mode.

Some Linux distributions do not require a password when entering single user mode. While this makes system administration somewhat easier, it is a considerable security problem. Make sure, that your system does not suffer from it.

For other security problems with running insecure programs under Linux or using poorly configured distributions, refer to the Usenet newsgroups, security mailing lists and choose a distribution whose manufacturer frequently releases security fixes. It is a fallacy to assume that the unavailability of patches implies the security and correctness of a software application; as a rule of thumb, a manufacturer who releases more patches than a different one, probably cares more about the security of your system than the latter. This also applies to operating systems other than Linux!

C.7. ETHERNET AND ITS PROTOCOLS

The ethernet is extremely vulnerable to attacks from malicious users. Anybody who can gain direct access to an ethernet segment, can easily monitor all traffic and inject forged data. This is very dangerous, because many protocols transfer data either un-encoded or in easily decipherable form. Also, authorization is often based on the assumption that the return address or a session id can be trusted, but this is no longer true if users gain unlimited access to the ethernet; it does not really matter if this access is achieved by having physical control over part of the network or by running a compromised or inherently insecure operating system. There are various attacks from machines that are not directly connected to your ethernet segment, but the majority of them can be prevented by installing and maintaining a properly configured firewall. For more information, you should regularly monitor security related newsgroups and mailinglists.

C.8. BOOTP/TFTP

BOOTP and TFTP offer almost no security whatsoever. They basically provide their information to anybody who asks and solely rely on the assumption that your network is configured to not make the server world-accessible. If you install BOOTP gateways, then this assumption is seriously violated. Also, TFTP server are usually accessible from just about everywhere. You can try to diminish the impact of this problem by blocking BOOTP and TFTP packets from leaving or entering your network segment, but this will never be a completely secure solution.

Thus you should always assume that all of the files that your BOOTP and TFTP server offer are world readable. They must not contain any sensitive data. Also, the TFTP daemon must be configured to only allow access to selected files. Running it in a chroot'd environment might be a very good idea.

The BOOTP protocol is vulnerable against somebody else impersonating as a BOOTP server. While security aware operating systems, prevent non-privileged users from starting their own BOOTP servers, other operating systems do not allow this. This means, if any of your users can launch an arbitrary program under an insecure operating system on an arbitrary machine connected to your ethernet segment, then they have full control over the BOOTP boot process.

C.9. NFS

While NFS is very convenient for installing diskless machines, it provides almost no security. Data is transmitted unencrypted and authorization is solely based on the identity of IP addresses. Anybody who can forge ethernet packets, has full access over any data that is available via NFS. While there are protocol extension that try to address these shortcomings, I am not aware of any solution for Linux based machines. This means, you have to assume that all exported filesystems are freely read- and writable. Bear this in mind when deciding which data you intend to export.

C.10. TELNET/RLOGIN

Telnet and rlogin do not usually come with any effective protection other than simple password schemes. Data and even the password is transmitted as plain text. There are commonly available programs that constantly monitor the network for packets that contains passwords. Fortunately, the security of these protocols can be vastly improved by replacing them with the Secure Shell protocoll (http://www.cs.hut.fi/ssh). Preferably, all telnet and rlogin servers and clients should be removed from all machines.

C.11. THE X WINDOW SYSTEM

X provides some security when run over a network, but the scope of it is limited and exploits can easily be devised. At the very least, you should make sure that the xauth protocol is used as opposed to the vastly inferior xhost protocol. A better solution is provided by routing all X connections through a secure shell session. This does not only provide more reliable authentication, but it also encrypts all data.

C.12. CONCLUSION

While this text cannot do more than barely scratch the surface, it should help you in locating some of the more vulnerable sub-systems of your networks and your computers. It does not aim for completeness, but if you think that there is a topic which should be mentioned or if you want to update an entry, then please to contact me.

D. VendorTag extensions, 28 April 2001

Markus Gutschke, gutschk AT math PERIOD uni-muenster PERIOD de with changes by Ken Yap, ken_yap AT users PERIOD sourceforge PERIOD net

This documentation has been written and is copyrighted 1996,97 by Markus Gutschke <gutschk AT math PERIOD uni-muenster PERIOD de>. You are free to distribute this file as long as you do not change its contents. I appreciate comments and will consider them in future revisions. If you have any questions, comments, or suggestions, please also send carbon copies of your e-mail message to both Ken Yap

<ken_yap AT users PERIOD sourceforge PERIOD net> and Gero Kuhlmann <gero AT gkminix PERIOD han PERIOD de>. I will happily include any of your extensions, but I would like to avoid the proliferation of different incompatible revisions of this document. If these conditions are a problem to you, then feel free to contact me.

RFC1533 allows for vendor-specific extensions to the BOOTP protocol. It defines that all tags in the range 128 thru 254 are set aside for site-specific extensions. Common implementations of DHCP daemons can assign arbitrary null-terminated character strings to these tags.

This is a list of the tags that are currently used by Etherboot. Tag 128 and 129 are handled by Etherboot. The other tags are handled by an external menu program, generated using mkelf-menu. This is changed behaviour from 5.0, where the menuing code was part of Etherboot. As a general rule, you should never fill out any of the other tags, unless you positively know that your BOOT-Prom supports the extension or ignores this particular tag:

TAG 128

this is a six-byte hexadecimal entry. The first four bytes have to be the magic number E4 45 74 68; if this magic cannot be found, then none of the other vendor tags are valid! The fifth byte is the major version number and the sixth byte is the minor version number of this protocol extension. The current version is 0.0; the BOOT-Prom can assume that incompatible changes increase the major version number. Mere extensions to the existing protocol, increase the minor version number. If the BOOT-Prom code has been written in a way that anticipates future extensions, then it is acceptable to honor the vendor tags even though, the minor version number does not match exactly. Before making a change, that requires updating the major version number, you should contact all of the persons that are listed at the top of this document.

TAG 129

the BOOT-Prom uses this tag for passing a user-provided command line to Linux. Anything in this tag will be appended to the kernel parameters already in the boot image created by mkelf.

TAG 160

a string that contains a colon separated list of "parameter=value" pairs. Currently, these parameters are only used to control the behavior of an interactive menu for selecting different boot images; future extensions are quite likely:

timeout

after this many seconds, the default image will be loaded. If no 'timeout' has been set, then the program will wait indefinitely.

default

either an integer 'n' in the range 0 thru 15 or 192 thru 207, selecting the default image. If 'n' is in the first range, it refers to the 'n'th menu entry; if it is in the later range, it refers to the entry with the tag number 'n'. This distinction is important, if the list of images contains gaps. If no value has been set, then the image with the lowest tag number will be the default image.

TAGS 161 thru 174

these tags are currently unused, but they should be allocated for purposes that resemble the function of tag 160.

TAG 175

the BOOT-Prom uses this tag for telling the DHCP server which NIC driver it is using. The DHCP server can then modify the "filename" field in the DHCPACK packet in order to direct the BOOT-Prom to download an appropriate kernel image.

TAG 176

the BOOT-Prom uses this tag for telling the loaded operating system, which of the entries in tags 192 to 207 has been selected by the user. As this tag is used internally, the DHCP daemon must not assign any value to it!

TAGS 177 thru 183

these tags are reserved for passing information from the BOOT-Prom to the boot image. Under no circumstances should the DHCP daemon assign any of these entries. The format of these parameters is yet to be discussed.

TAGS 184 thru 191

up to eight zero-terminated character strings can be used for displaying a "message of the day". If you need to display more than eight lines, you can embed suitable CR/LF pairs. If you fully exploit that feature, you can display a complete 80x24 screen of information, but you should be aware that subsequent output might scroll part of a long message.

The BOOT-Prom can optionally be configured to interpret some ANSI escape sequences.

The ANSI emulation currently knows about these commands:

```
Display attributes

Code          Effect
<esc>[0m      normal text
<esc>[1m      high-intensity on
<esc>[21m     high-intensity off
<esc>[5m      blinking on
<esc>[25m     blinking off
<esc>[7m      reverse video on
<esc>[27m     reverse video off
<esc>[3xm     set foreground color:
<esc>[4xm     set background color. x can be:
 0 - black                 4 - blue
 1 - red                   5 - magenta
 2 - green                 6 - cyan
 3 - yellow                7 - white
<esc>[=xh     set video mode
 0 - 40x25 mono    (text)  13 - 40x25 16colors (gfx)
 1 - 40x25 16colors (text) 14 - 80x25 16colors (gfx)
 2 - 80x25 mono    (text)  15 - 80x25 mono     (gfx)
 3 - 80x25 16colors (text) 16 - 80x25 16colors (gfx)
 4 - 40x25 4colors  (gfx)  17 - 80x30 mono     (gfx)
```

```
   5 - 40x25 mono      (gfx)    18 - 80x30 16colors (gfx)
   6 - 80x25 mono      (gfx)    19 - 40x25 256colors(gfx)
```

Cursor control

```
  Code          Effect
  <esc>[r;cH    move cursor to row r and column c
  <esc>[r;cf    move cursor to row r and column c
  <esc>[rA      move cursor up r rows
  <esc>[rB      move cursor down r rows
  <esc>[cC      move cursor right c columns
  <esc>[cD      move cursor left c columns
  <esc>[?7l     turn off line wrap
  <esc>[?7h     turn on line wrap
  <esc>[J       clear screen and home cursor
  <esc>[K       clear to end of line
  <esc>[s       save the cursor position
  <esc>[u       return cursor to saved position
```

Extended features

```
  Code          Effect
  <esc>[a;b;c;d+<data>
  draw pixel data. Use  one byte  per  pixel.
  Colors are encoded as  shown above.  In text
  mode, graphics is approximated by outputting
  suitable   characters. Parameters   differ
  depending   on  the number   of parameters
  passed:
    cnt
      "cnt"  data  bytes follow. They  will be
      drawn to  the right of the last graphics
      position.
    rle;col
      the next  "rle"  pixels  have  the  value
      "col". They will  be drawn to  the right
      of the last  graphics position.  No data
      bytes follow.
    x;y;cnt
      "cnt" data  bytes  follow. They  will be
      drawn relative to the top left corner of
      the text cursor with an offset of (x/y).
    x;y;rle;col
      the next "rle"  pixels have  the  value
      "col".  They will  be drawn  relative to
      the top  left corner of the  text cursor
      with an offset  of (x/y).  No data bytes
      follow
  you usually  do not  have  to enter  these
  values manually, but you  should use  a tool
  such as  "ppmtoansi" which  is  shipped with
```

```
your BOOT-Prom or available from the contrib
directory of the "etherboot" package.
<esc>[a;b;c;d-<data>
same as above,  but pack pixels into  three
bits. The first pixel is stored in the three
most  significant  bits  of  the  first  data
byte.
```

Note that you usually have to specify any control characters directly (rather than in hex form) in your `/etc/dhcpd.conf` file.

TAGS 192 thru 207

these tags define all of the valid boot images and override any settings that are given with the filename field in your `/etc/dhcpd.conf`. It is allowed to leave gaps in the list. This has an impact on how the default image will be selected.

All entries are of the form

```
label:server:gateway:filename:passwd:flags:cmdline
```

For future extensibility, it is permitted to append an arbitrary amount of other colon separated entries as long as the limit of 255 characters per tag is not exceeded. Non-existent entries can be left empty. This means that the default value for this particular entry will be used. Trailing colons can be omitted.

label

this is the text string that is displayed to the user. It can contain arbitrary characters, except for a colon. Embedding arbitrary control characters is not recommended, but you might be able to include ANSI escape sequences (if enabled in the ROM) for changing text attributes as long as you restore the attributes at the end of the string. It probably does not make very much sense to leave this entry empty.

server

(This field is no longer supported. Its contents will be ignored.)

gateway

(This field is no longer supported. Its contents will be ignored.)

filename

name of the boot image that has to be loaded by TFTP. If this entry is omitted, then the machine boots locally from disk. If enabled in the BOOT-Prom, you can specify pseudo-filenames for booting from a local blockdevice (floppy, harddisk, ...); these filenames have to match the pattern "/dev/[fh]d*". If the BOOT-Prom does not have support for these pseudo-filenames, you can still boot from blockdevices by storing an boot image as generated by mknbi-blkdev under the name of the desired blockdevice (symbolic links will do). In Etherboot 4.6.2 and later, a - in this field means use the filename specified in the BOOTP/DHCP reply. This saves on menu size.

passwd

   MD5 message digest of the password. If this entry is omitted, then no password is required for loading this image. Support for passwords is optional and might not be compiled into the ROM image. For generating the MD5 message digest, you can use freely available tools such as "md5sum". C.f. the flags entry for controlling the behavior of passwords.

flags

   flags are used for controlling some aspects of how the BOOT-Prom code behaves. All flags are a string of decimal digits followed by a letter; multiple flags can be concatenated. If this entry is omitted, then a default value of "1i1p" is assumed. Currently, these flags are defined:

   0i

      booting this image does not require a password; the contents of the password entry is ignored unless some other feature (such as the flag "2p") requires it.

   1i

      booting this image requires a password. If the password entry is omitted, or no password support is available in the BOOT-Prom, then this flag is ignored.

   0p

      the user cannot enter a command line for passing parameters to the loaded image, even if this feature has been enabled when compiling the BOOT-Prom. N.B. this does not affect the cmdline entry as described below!

   1p

      the user does not get prompted for passing parameters to the loaded image, but he can explicitly request the prompt (e.g. by pressing a modifier key while selecting an image from the menu). If the password entry is not omitted, then the password has to be entered. Both parameter passing and password validation can be disabled when compiling the BOOT-Prom.

   2p

      the user always gets prompted for passing parameters to the loaded image. If the password entry is present and password support has been enabled in the BOOT-Prom, then the password has to be entered.

   3p

      the user always gets prompted for passing parameters to the loaded image. No password is required.

cmdline

   the contents of this entry is appended to the end of tag 129 from the DHCP record, provided tag 129 exists. This feature is unaffected by the "p" flags. Passing parameters currently does

not make sense for any operating system other than Linux and is silently ignored for other operating systems. As it is not legal to enter colons as part of an entry, you have to escape them by writing "~c" instead. This also means, that all tilde characters have to be escaped by writing "~~". As some DHCP daemons do not allow for entering a backslash in a character string, the escape sequence "~b" inserts a backslash character. Currently, all other escape sequences are undefined.

(This example has not yet been translated to DHCPD syntax, sorry.) For demonstration purposes, I attached an annotated excerpt from my "bootptab" illustrating some of these techniques. In the following the character sequence ESC should be replaced by the ASCII escape character. Also the 8-bit characters have been changed to 7-bit approximations due to SGML tool limitations. To get the original codes, use this table: top left corner, char 201 (decimal); horizontal bar, char 205; top right corner, char 187; vertical bar, char 186; bottom left corner, char 200; bottom right corner, char 188.

```
#
# The MOTD (message of the day) can contain arbitrary characters, that
# the PC is capable of displaying.  Here we use 8bit characters for
# drawing a border around the message; also we change the foreground
# color to red (this assumes that the BOOT Prom has support for ANSI
# escape sequences):
#
.motd:\
        :T184="ESC[31m":\
        :T185="+----------------------------------------------------+":\
        :T186="| This is an experimental release of the new BOOT-Prom |":\
        :T187="| code.  It supports a couple of  non-standard  vendor |":\
        :T188="| extensions.                                        |":\
        :T189="+----------------------------------------------------+":\
        :T190="ESC[37m":


#
# Alternatively, the MOTD can be stored in an external file; this
# requires that you enabled ANSI support in the BOOT Prom!  For more
# advanced configurability, you should explore the feature of the
# patched tftpd daemon (as available in the contrib directory) to
# execute shell scripts and use the output as the file contents.
#
# .motd:\
#        :T184="ESC[31mLoading message of the day...ESC[37mESC['/etc/motd'#":
#


#
# We use the "template" feature of modern versions of "bootp" in order
# to group common entries. Unfortunately, you cannot use more than one
# "tc" entry per (pseudo-) host.  Pseudo hostnames should begin with a
# leading period character.
#
# All entries are kept as generic as possible.  This ensures that they
```

```
# will keep working even when the network topology is changed. Leaving
# the server IP address and the gateway IP address empty, should
# ensure that booting works even when we go thru "bootpgw" gateways.
#
# "Technicolor" special effects are achieved by changing the
# foreground text color of the individual labels. You could even embed
# small icons after switching to graphics mode. C.f. "linux-logo.ansi"
# for an example; this will not work, if your BOOT Prom does not have
# support for ANSI escape sequences.
#
# Passing boot-time parameters to the Linux kernel, requires the
# password "Penguin".
#
# Booting from the local disk requires the password "Joshua". If the
# BOOT-Prom does not have support for booting from local block devices
# (floppy, harddrive, ...), then you can either omit the filename
# (c.f. README.Security for potential security problems) or your TFTP
# server has to provide a boot image that has been generated by
# mknbi-blkdev.
#
.imagemenu:\
        :tc=.motd:\
        :T128=E44574680000:\
        :T160="timeout=30:default=207:":\
        :T192="ESC[32mLinux 2.0.27ESC[37m:::/tftpdir/image-linux:99625fa1cac27bb6a2b33b7
        :T193="ESC[33mDOS 6.2ESC[37m:::/tftpdir/image-dos":\
        :T207="ESC[34mLocal DiskESC[37m:::/dev/hda:85b103482a20682da703aa388933a6d8":

#
# When using more than a handful of vendor parameters, we have to
# specify an extension file "ef". If you are very careful, then it is
# possible to use one extension file for several hosts. Don't forget
# to run "bootpef" after editing the "bootptab", and make sure that
# you "chroot" into the proper directory, if applicable.
#
.default:\
        :tc=.imagemenu:\
        :bf=tftpdir/image-rom:\
        :hd=:\
        :ht=ethernet:\
        :sm=255.255.255.0:\
        :vm=auto:\
        :ef=extension.bootp:

#
# Ideally, hosts differ only with respect to their ethernet hardware
# ID and IP number. We let the "bootpd" look up the correct IP number.
#
thalamus:tc=.default:ha=00400529C11B:ip=thalamus:
cortex: :tc=.default:ha=0000C0531A24:ip=cortex:
```

Here is an example DHCP specification that uses an external menu program. It also illustrates the use of
- in the filename portion of the menu options to eliminate needless repetition.

```
# per host setup
host 192.168.40.203 {

        # MAC and IP addresses
        hardware ethernet  00:60:08:0d:a9:84;
        fixed-address 192.168.40.203;

        # default file to boot, common append options, default menu selection and timeout
        filename "/tftpboot/menu.nb";
        option option-128 E4:45:74:68:00:00;
        option option-129 "ramdisk_size=16000 vga=6";
        option option-160 "timeout=10:default=192";
        option option-184 "^[['/tftpboot/thinlinux/1.0-alpha-025/motd'#";

        # menu selections, specify filename or "-" to use default filename specified above
        option option-192 "test1:::-:::nfs=xterm";
        option option-193 "test2:::-:::nfs=shell";
        option option-194 "test3:::-:::nfs=other";
    }
```

For ISC DHCPD 3.0 Beta 2 Patchlevel 18 and above the syntax above is no longer accepted and a new syntax is in operation according to this note in the Changelog:

```
Use unparsable names for unknown options. WARNING: this will break any
configuration files that use the option-nnn convention. If you want to
continue to use this convention for some options, please be sure to
write a definition, like this:

option option-nnn code nnn = string;

You can use a descriptive name instead of option-nnn if you like.
```